

FACULDADE METODISTA GRANBRY – FMG
CURSO SISTEMAS DE INFORMAÇÃO

TESTE DE SOFTWARE

**FREDERICO CASSEMIRO
MATHEUS PERES DE ARAÚJO**

JUIZ DE FORA – 2018

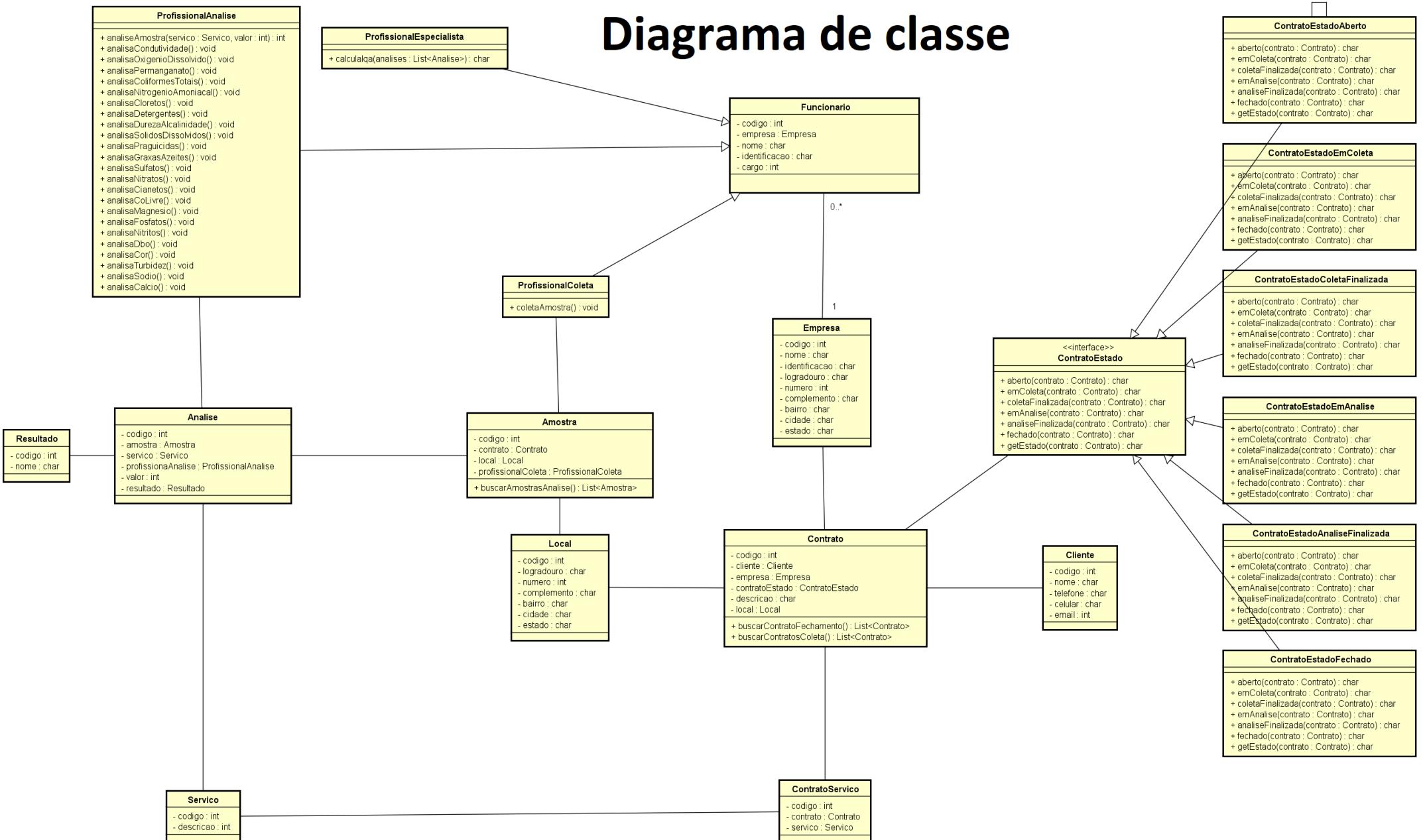
1 – INTRODUÇÃO

Para realização do trabalho, foi desenvolvido um sistema que permite o controle da qualidade da água. O sistema pode ser descrito em 4 etapas: contrato, coleta, análise e entrega. Na primeira etapa, contrato, é configurado os objetivos que serão realizados, seu contratante e a empresa responsável pelo projeto em questão. A próxima etapa, coleta, declara o processo operacional onde um dos funcionários de coleta realiza a coleta de amostras para a posterior análise. A terceira etapa, análise, descreve e aplica nas amostras coletadas todos os indicadores que foram definidos no contrato. A última etapa, entrega, tem como objetivo validar as informações e finalizar o contrato com o resultado do serviço.

2 – DIAGRAMA DE CLASSE

Segue abaixo o diagrama desenvolvido para representar o sistema.

Diagrama de classe



3 – ANÁLISE DAS CLASSES

Através do diagrama de classe foi realizado uma análise das classes com o objetivo de identificar as de maior complexidade. Para tal tarefa foram utilizadas as métricas de Chidamber & Kemerer e Lorenz & Kidd. Segue abaixo o resultado encontrado.

Classe/Métricas	WMC	DIT	NOC	CBO	CS	NOO	NOA
Amostra	1	0	0	3	5	0	0
Analise	0	0	0	4	6	0	0
Cliente	0	0	0	1	5	0	0
Contrato	2	0	0	5	8	0	0
ContratoEstado	7	0	6	1	7	0	0
ContratoEstadoAberto	7	1	0	0	14	7	0
ContratoEstadoAnaliseFinalizada	7	1	0	0	14	7	0
ContratoEstadoColetaFinalizada	7	1	0	0	14	7	0
ContratoEstadoEmAnalise	7	1	0	0	14	7	0
ContratoEstadoEmColeta	7	1	0	0	14	7	0
ContratoEstadoFechado	7	1	0	0	14	7	0
ContratoServico	0	0	0	2	3	0	0
Empresa	0	0	0	2	9	0	0
Funcionario	0	0	3	1	5	0	0
Local	0	0	0	2	7	0	0
ProfissionalAnalise	25	1	0	1	30	0	1
ProfissionalColeta	1	1	0	1	6	0	1
ProfissionalEspecialista	1	1	0	1	6	0	1
Resultado	0	0	0	1	2	0	0
Servico	0	0	0	2	3	0	0

Utilizando o método de Pareto podemos definir que em 20% das classes podemos resolver 80% dos problemas. Dentre as 20 classes, foram escolhidas 4 para execução de testes.

ProfissionalAnalise é a classe que tem maior WMC e CS.

Contrato é a classe que possui maior número de ligações dentro do sistema.

ContratoEstadoAberto; ContratoEstadoEmAnalise são duas das classes estado que estão ligadas a classe contrato e possui métodos que são muito utilizados.

4 – ANÁLISE DOS MÉTODOS

Realizando o cálculo de complexidade ciclomática de McCabe dos métodos foi encontrado um método com complexidade igual a 24 e outros vinte e quatro métodos com complexidade igual a 11.

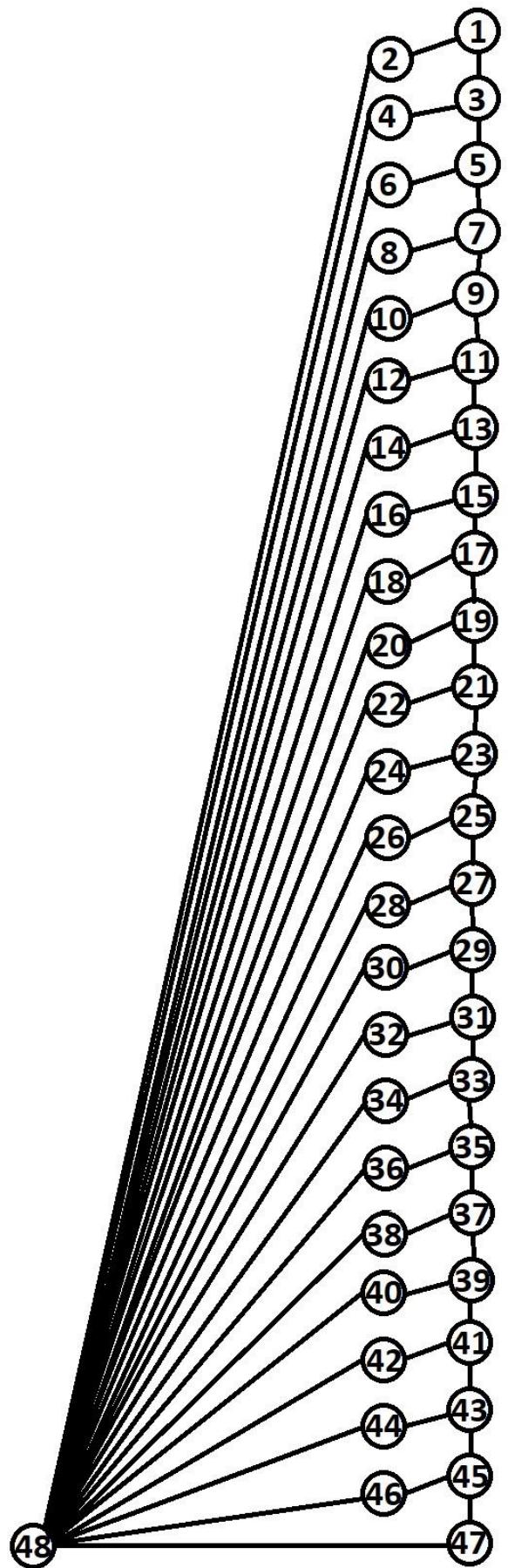
4.1 – Apresentação dos métodos

Segue abaixo os métodos de maior complexidade.

Método: analiseAmostra

Complexidade: 24

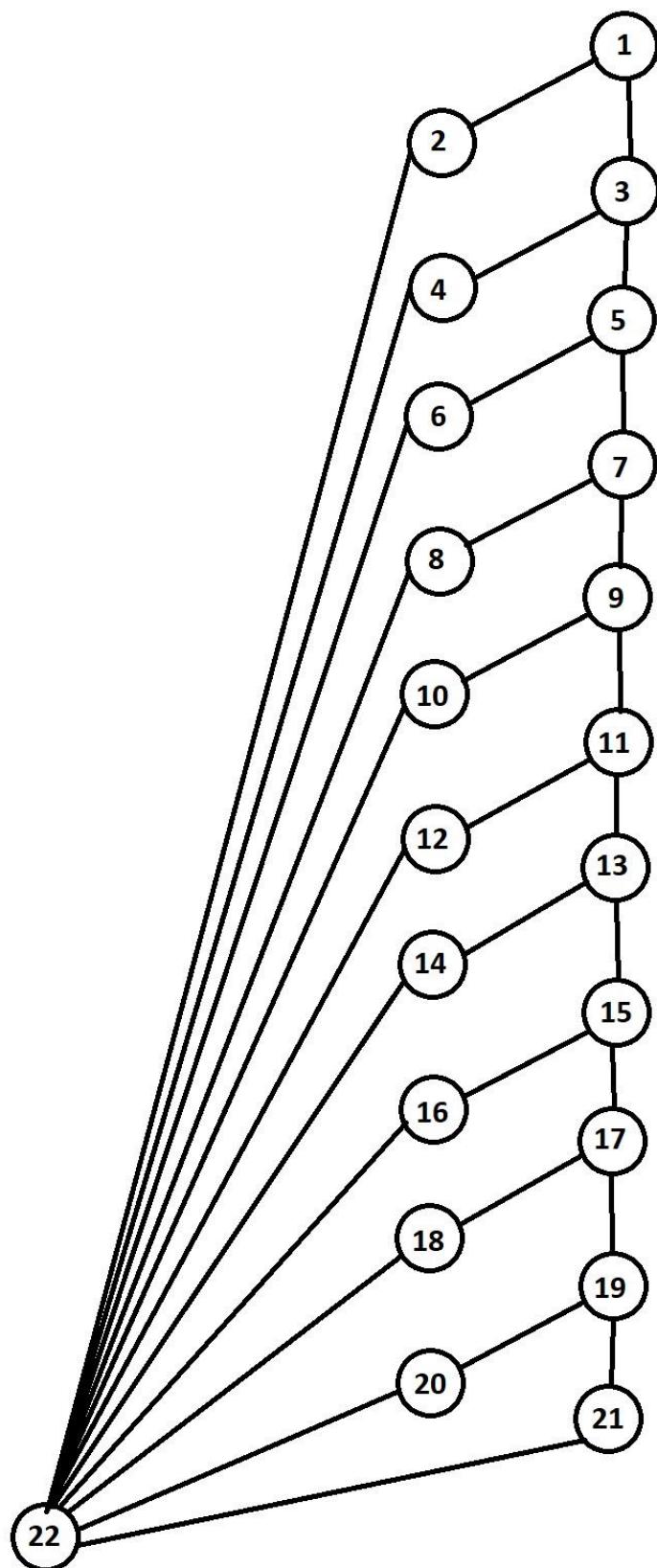
```
public Integer analiseAmostra(double codigoServico, double valor){  
  
    if(codigoServico == 1){  
        return analisaPh(valor);  
    }else if(codigoServico == 2){  
        return analisaCondutividade(valor);  
    }else if(codigoServico == 3){  
        return analisaOxigenioDissolvido(valor);  
    }else if(codigoServico == 4){  
        return analisaReducaoPermanganato(valor);  
    }else if(codigoServico == 5){  
        return analisaColiformesTotais(valor);  
    }else if(codigoServico == 6){  
        return analisaNitrogenioAmoniacal(valor);  
    }else if(codigoServico == 7){  
        return analisaCloreto(valor);  
    }else if(codigoServico == 8){  
        return analisaDetergentes(valor);  
    }else if(codigoServico == 9){  
        return analisaDurezaAlcalinidade(valor);  
    }else if(codigoServico == 10){  
        return analisaSolidosDissolvidos(valor);  
    }else if(codigoServico == 11){  
        return analisaPraquicidas(valor);  
    }else if(codigoServico == 12){  
        return analisaGraxasAzeites(valor);  
    }else if(codigoServico == 13){  
        return analisaSulfatos(valor);  
    }else if(codigoServico == 14){  
        return analisaNitratos(valor);  
    }else if(codigoServico == 15){  
        return analisaCianetos(valor);  
    }else if(codigoServico == 16){  
        return analisaCoLivre(valor);  
    }else if(codigoServico == 17){  
        return analisaMagnesio(valor);  
    }else if(codigoServico == 18){  
        return analisaFosfatos(valor);  
    }else if(codigoServico == 19){  
        return analisaNitritos(valor);  
    }else if(codigoServico == 20){  
        return analisaDbo(valor);  
    }else if(codigoServico == 21){  
        return analisaCor(valor);  
    }else if(codigoServico == 22){  
        return analisaTurbidez(valor);  
    }else if(codigoServico == 23){  
        return analisaSodio(valor);  
    }else{  
        return analisaCalcio(valor);  
    }  
}
```



Método: analisePh

Complexidade: 11

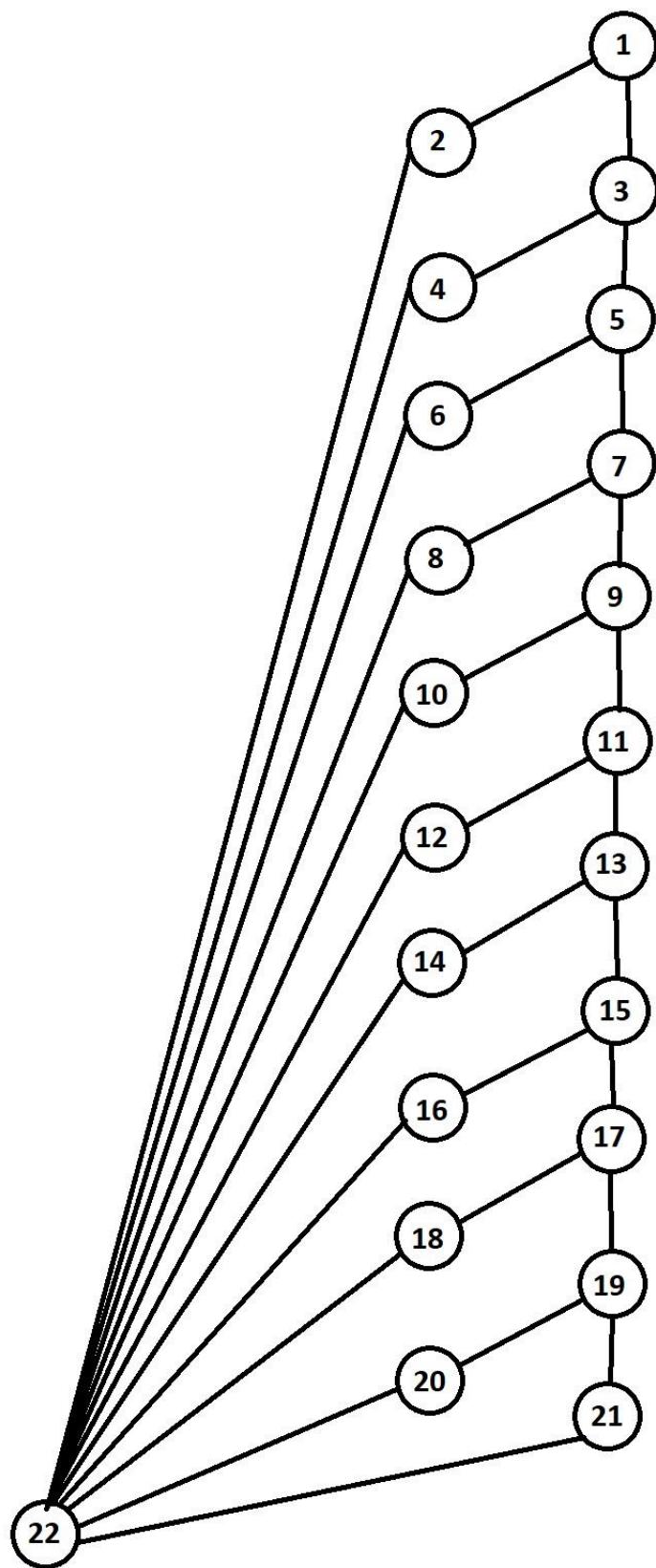
```
public Integer analisaPh(double valor){  
  
    if(valor == 1){  
        return 1;  
    }else if(valor == 2){  
        return 2;  
    }else if(valor == 3){  
        return 3;  
    }else if(valor == 4){  
        return 4;  
    }else if(valor == 5){  
        return 5;  
    }else if(valor == 6){  
        return 6;  
    }else if(valor == 7){  
        return 7;  
    }else if(valor == 8){  
        return 8;  
    }else if(valor == 9){  
        return 9;  
    }else if(valor == 10){  
        return 10;  
    }else{  
        return 11;  
    }  
}
```



Método: analiseCondutividade

Complexidade: 11

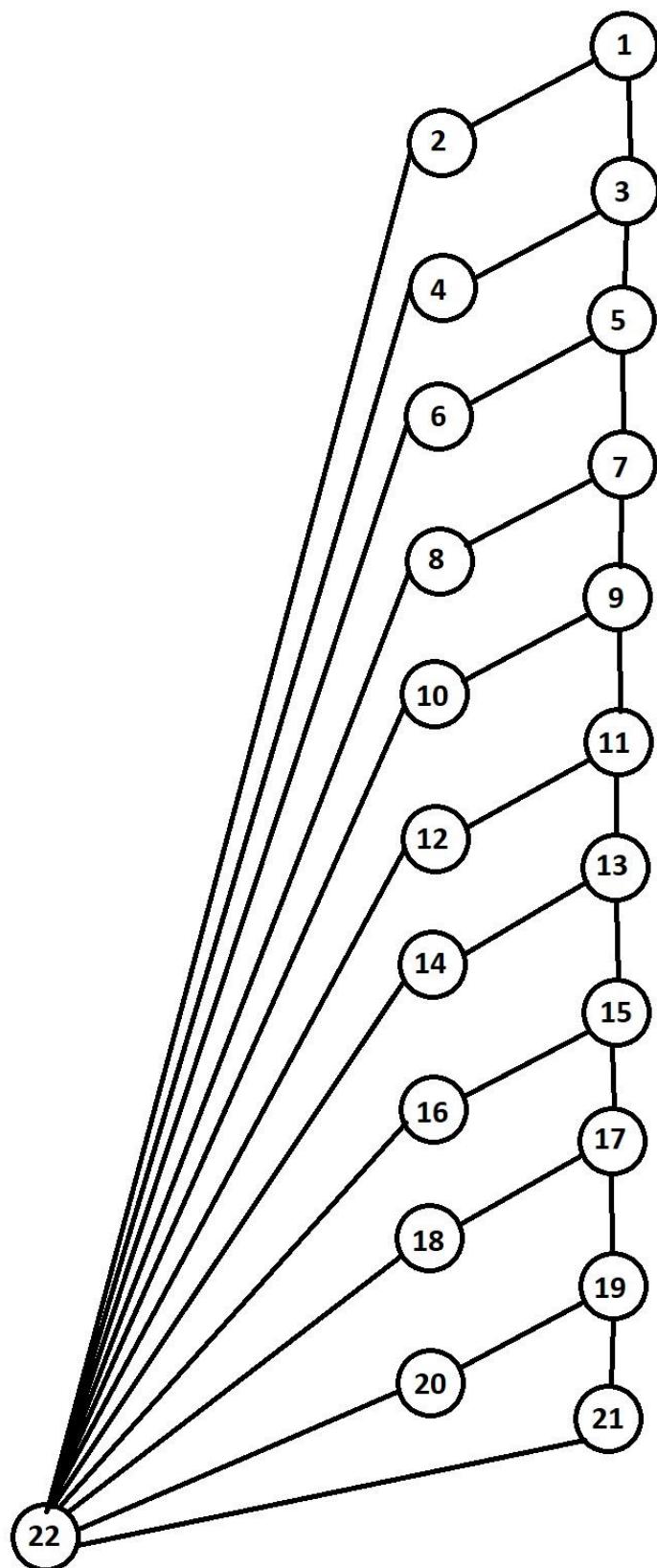
```
public Integer analisaCondutividade(double valor){  
    if(valor > 16000){  
        return 1;  
    }else if(valor > 12000){  
        return 2;  
    }else if(valor > 8000){  
        return 3;  
    }else if(valor > 5000){  
        return 4;  
    }else if(valor > 3000){  
        return 5;  
    }else if(valor > 2500){  
        return 6;  
    }else if(valor > 2000){  
        return 7;  
    }else if(valor > 1500){  
        return 8;  
    }else if(valor > 1250){  
        return 9;  
    }else if(valor >= 750){  
        return 10;  
    }else{  
        return 11;  
    }  
}
```



Método: analisaOxigenioDissolvido

Complexidade: 11

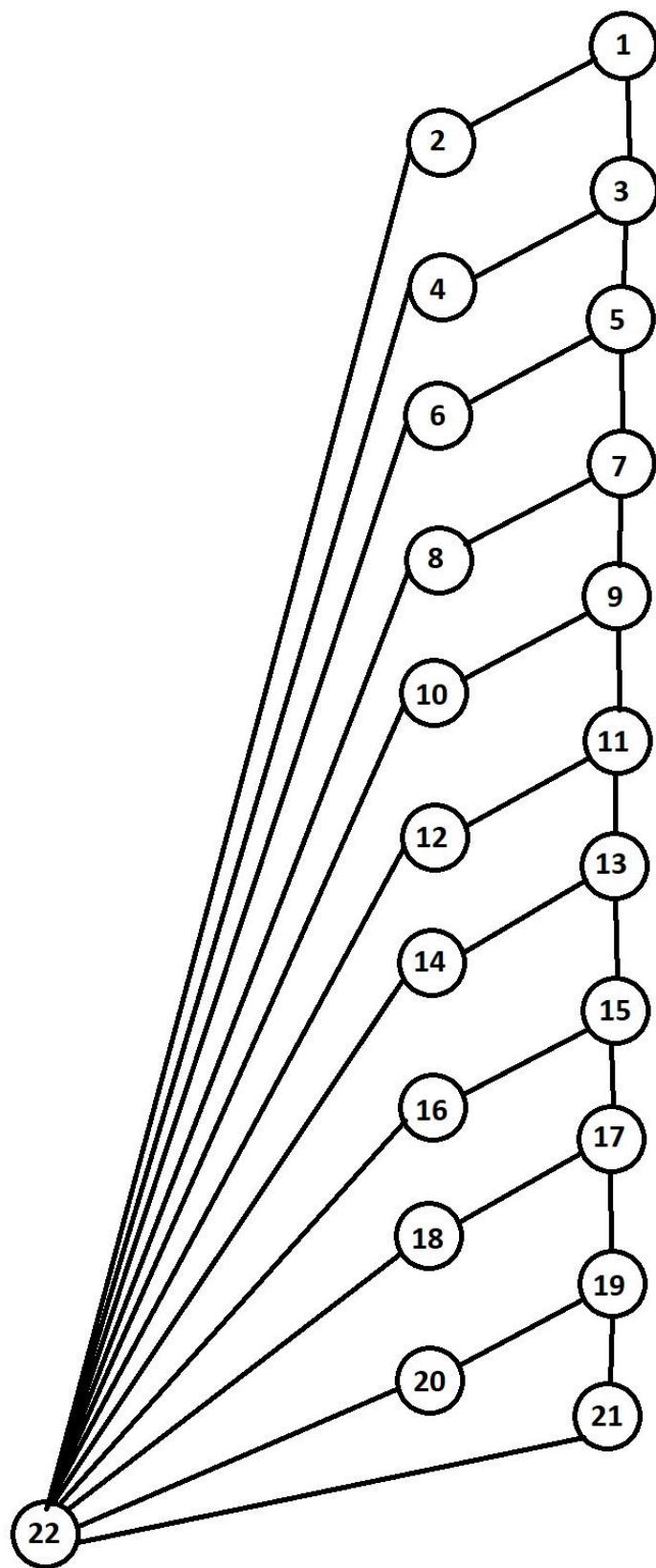
```
public Integer analisaOxigenioDissolvido(double valor){  
  
    if(valor < 1){  
        return 1;  
    }else if(valor < 2){  
        return 2;  
    }else if(valor < 3){  
        return 3;  
    }else if(valor < 3.5){  
        return 4;  
    }else if(valor < 4){  
        return 5;  
    }else if(valor < 5){  
        return 6;  
    }else if(valor < 6){  
        return 7;  
    }else if(valor < 6.5){  
        return 8;  
    }else if(valor < 7){  
        return 9;  
    }else if(valor < 7.5){  
        return 10;  
    }else{  
        return 11;  
    }  
  
}
```



Método: analisaReducaoPermanganato

Complexidade: 11

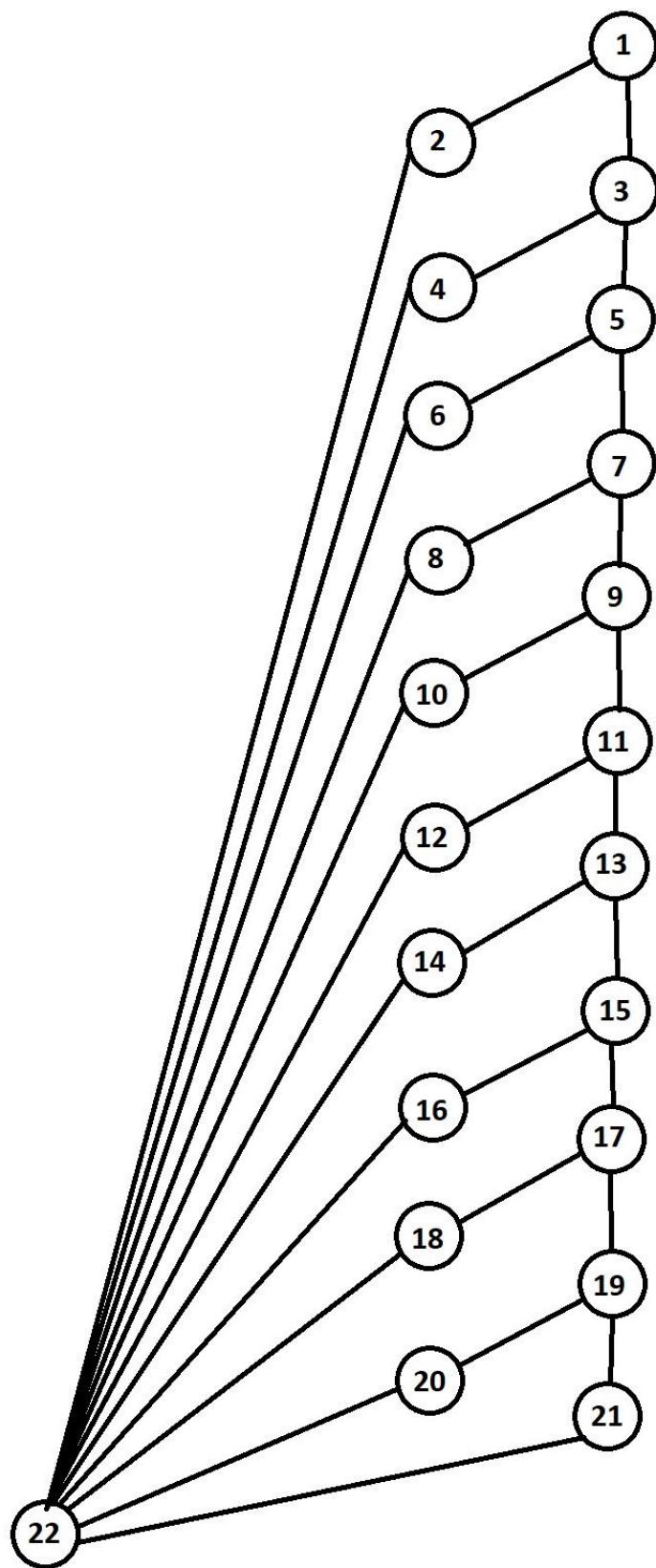
```
public Integer analisaReducaoPermanganato(double valor){  
  
    if(valor > 15){  
        return 1;  
    }else if(valor > 12){  
        return 2;  
    }else if(valor > 10){  
        return 3;  
    }else if(valor > 8){  
        return 4;  
    }else if(valor > 6){  
        return 5;  
    }else if(valor > 5){  
        return 6;  
    }else if(valor > 4){  
        return 7;  
    }else if(valor > 3){  
        return 8;  
    }else if(valor > 2){  
        return 9;  
    }else if(valor > 0.5){  
        return 10;  
    }else{  
        return 11;  
    }  
  
}
```



Método: analisaColiformesTotais

Complexidade: 11

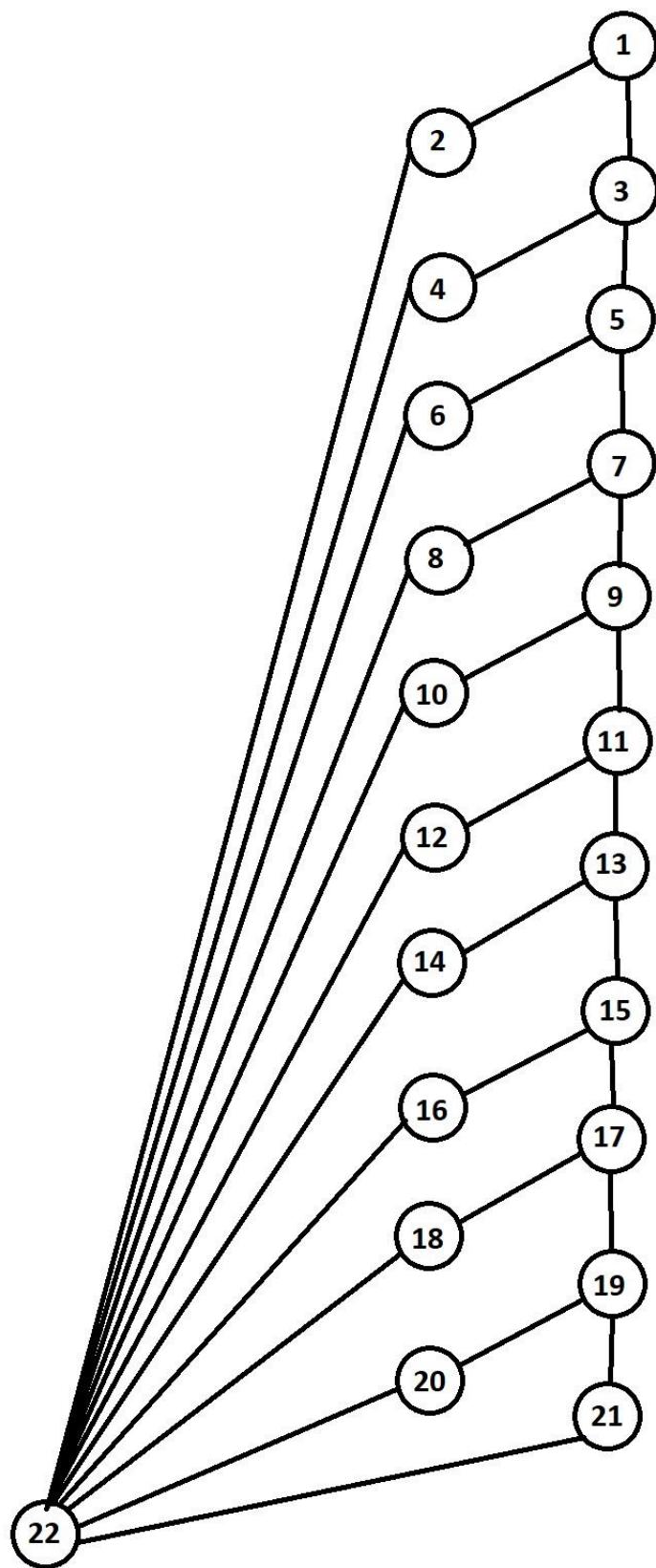
```
public Integer analisaColiformesTotais(double valor){  
  
    if(valor > 14000){  
        return 1;  
    }else if(valor > 10000){  
        return 2;  
    }else if(valor > 7000){  
        return 3;  
    }else if(valor > 5000){  
        return 4;  
    }else if(valor > 4000){  
        return 5;  
    }else if(valor > 3000){  
        return 6;  
    }else if(valor > 2000){  
        return 7;  
    }else if(valor > 1500){  
        return 8;  
    }else if(valor > 1000){  
        return 9;  
    }else if(valor > 50){  
        return 10;  
    }else{  
        return 11;  
    }  
}
```



Método: analisaNitrogenioAmoniacal

Complexidade: 11

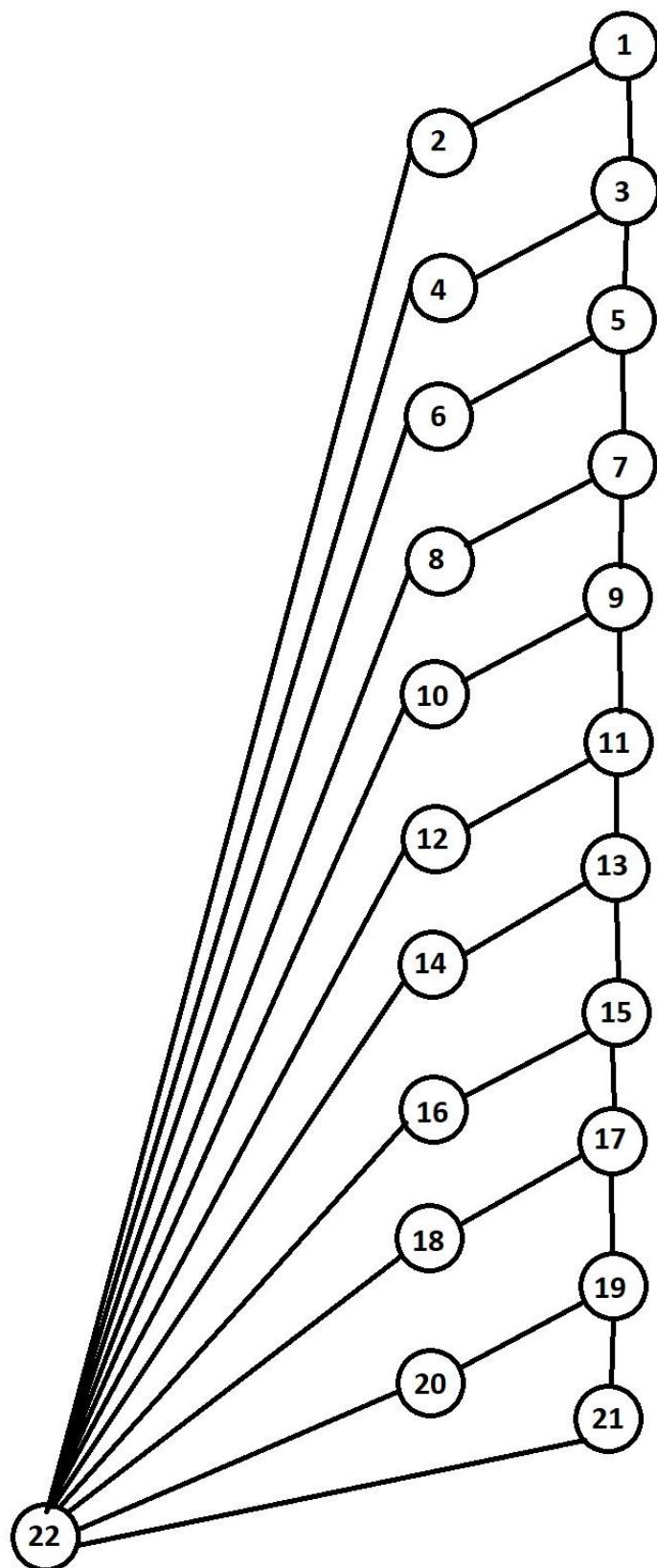
```
public Integer analisaNitrogenioAmoniacal(double valor){  
  
    if(valor > 1.25){  
        return 1;  
    }else if(valor > 1){  
        return 2;  
    }else if(valor > 0.75){  
        return 3;  
    }else if(valor > 0.5){  
        return 4;  
    }else if(valor > 0.4){  
        return 5;  
    }else if(valor > 0.3){  
        return 6;  
    }else if(valor > 0.2){  
        return 7;  
    }else if(valor > 0.1){  
        return 8;  
    }else if(valor > 0.05){  
        return 9;  
    }else if(valor > 0){  
        return 10;  
    }else{  
        return 11;  
    }  
}
```



Método: analisaCloretos

Complexidade: 11

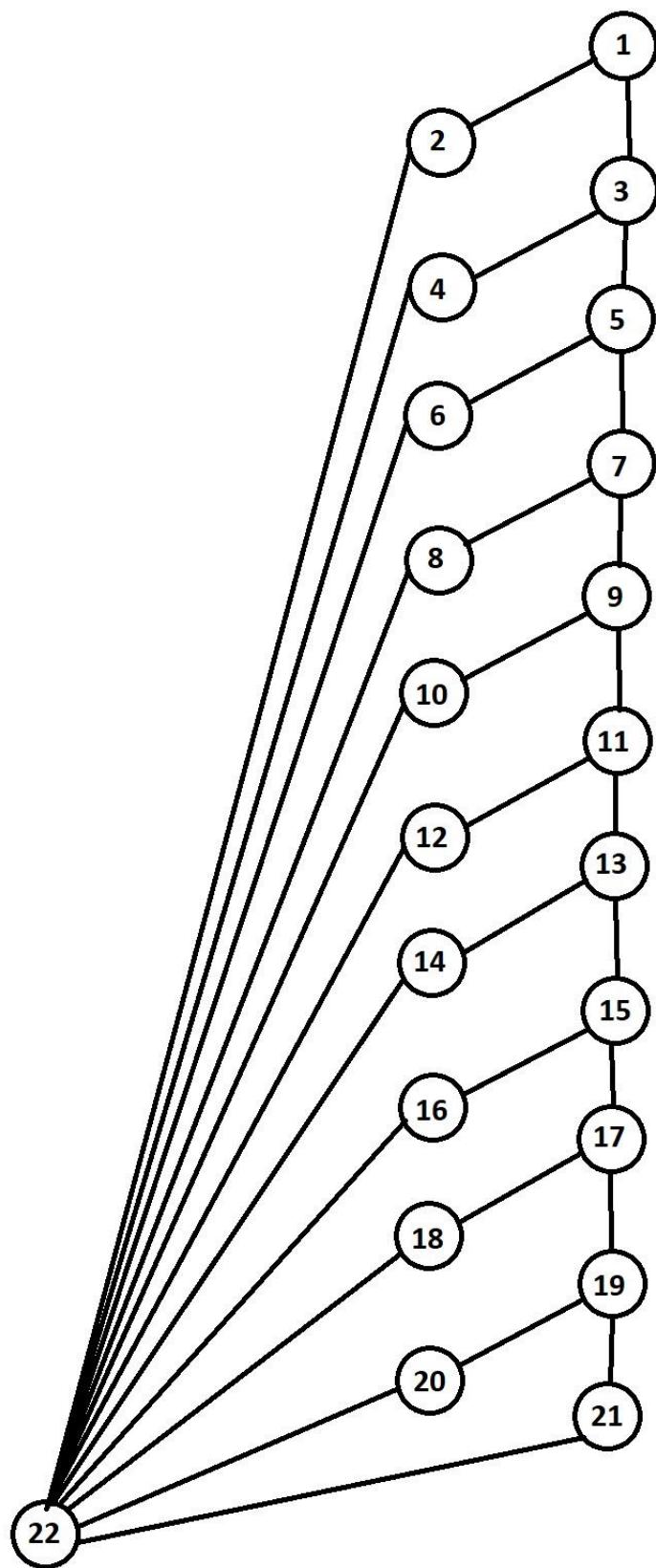
```
public Integer analisaCloretos(double valor){  
  
    if(valor > 1500){  
        return 1;  
    }else if(valor > 1000){  
        return 2;  
    }else if(valor > 700){  
        return 3;  
    }else if(valor > 500){  
        return 4;  
    }else if(valor > 300){  
        return 5;  
    }else if(valor > 200){  
        return 6;  
    }else if(valor > 150){  
        return 7;  
    }else if(valor > 100){  
        return 8;  
    }else if(valor > 50){  
        return 9;  
    }else if(valor > 0){  
        return 10;  
    }else{  
        return 11;  
    }  
}
```



Método: analisaDetergentes

Complexidade: 11

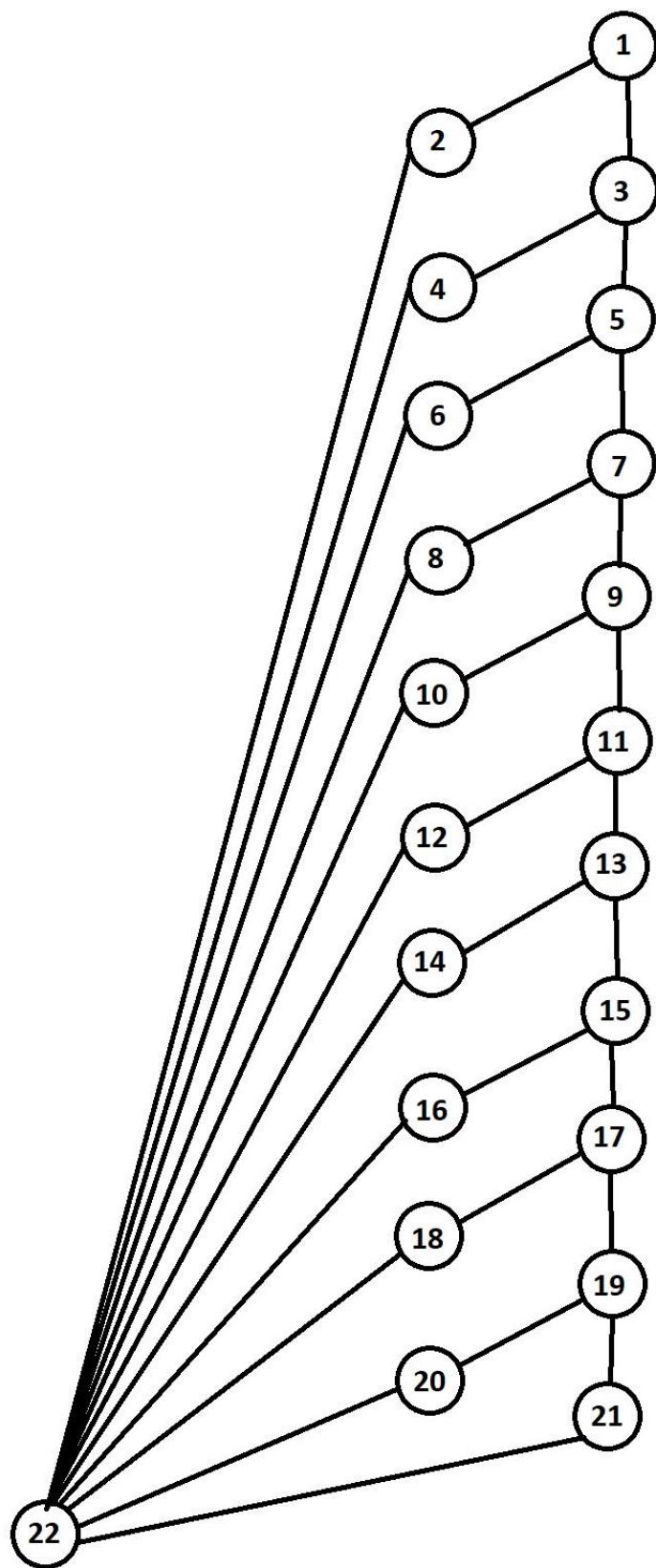
```
public Integer analisaDetergentes(double valor){  
  
    if(valor > 3){  
        return 1;  
    }else if(valor > 2){  
        return 2;  
    }else if(valor > 1.5){  
        return 3;  
    }else if(valor > 1){  
        return 4;  
    }else if(valor > 0.75){  
        return 5;  
    }else if(valor > 0.50){  
        return 6;  
    }else if(valor > 0.25){  
        return 7;  
    }else if(valor > 0.10){  
        return 8;  
    }else if(valor > 0.06){  
        return 9;  
    }else if(valor > 0){  
        return 10;  
    }else{  
        return 11;  
    }  
}
```



Método: analisaDurezaAlcalinidade

Complexidade: 11

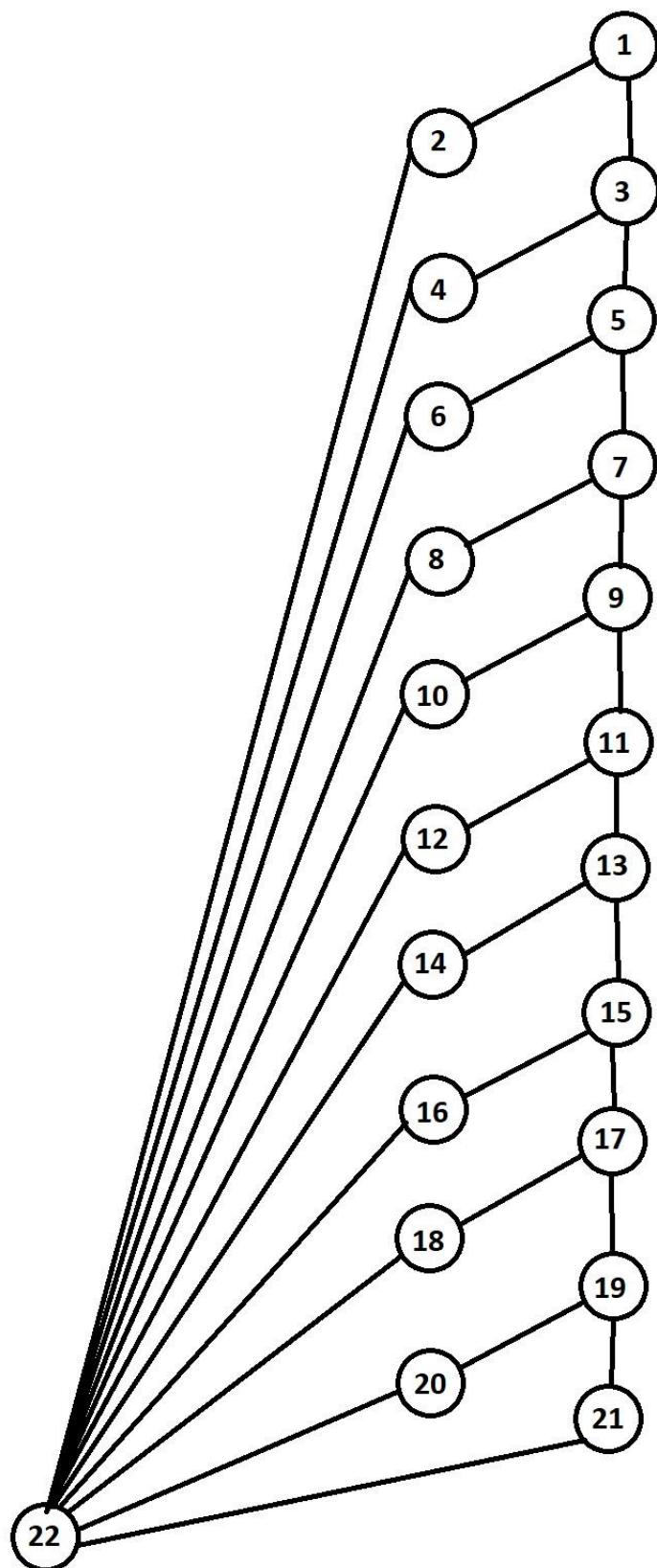
```
public Integer analisaDurezaAlcalinidade(double valor){  
  
    if(valor > 1500){  
        return 1;  
    }else if(valor > 1000){  
        return 2;  
    }else if(valor > 800){  
        return 3;  
    }else if(valor > 600){  
        return 4;  
    }else if(valor > 500){  
        return 5;  
    }else if(valor > 400){  
        return 6;  
    }else if(valor > 300){  
        return 7;  
    }else if(valor > 200){  
        return 8;  
    }else if(valor > 100){  
        return 9;  
    }else if(valor > 25){  
        return 10;  
    }else{  
        return 11;  
    }  
}
```



Método: analisaSolidosDissolvidos

Complexidade: 11

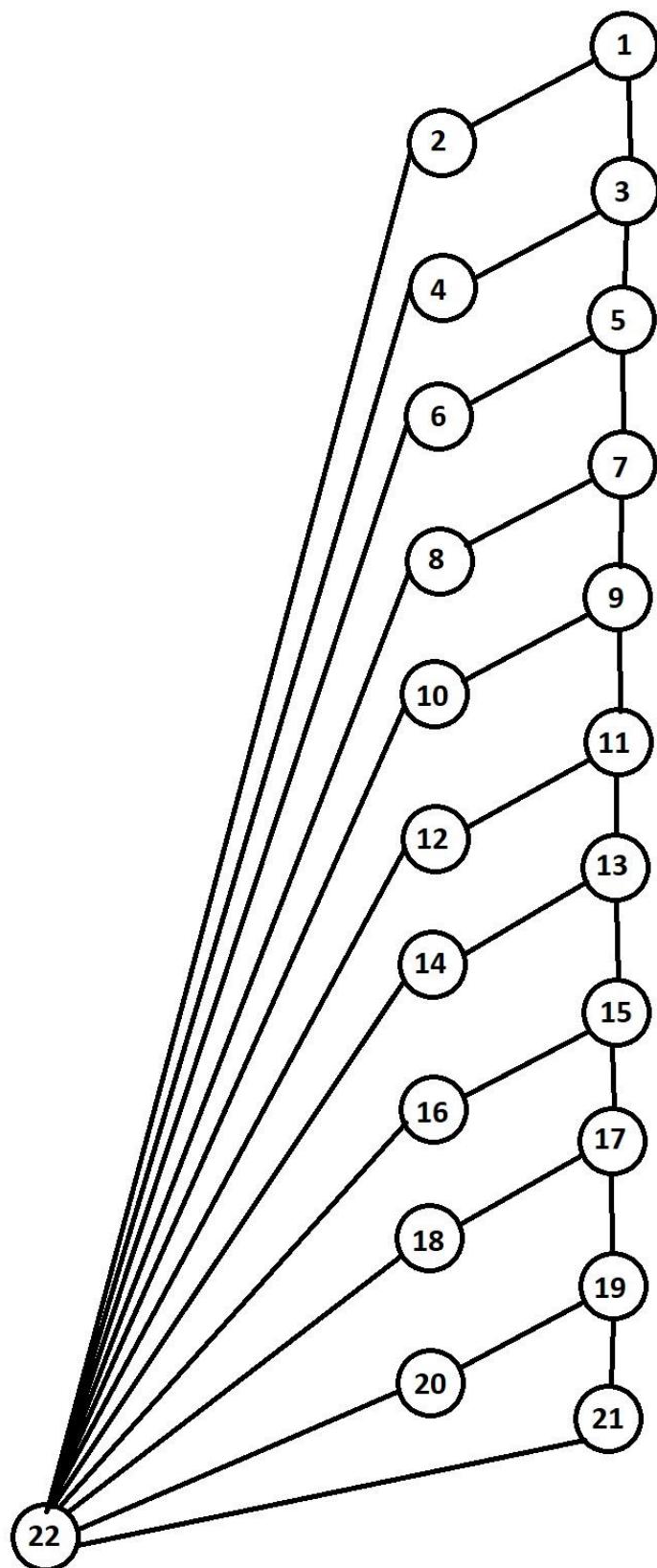
```
public Integer analisaSolidosDissolvidos(double valor){  
  
    if(valor > 20000){  
        return 1;  
    }else if(valor > 10000){  
        return 2;  
    }else if(valor > 5000){  
        return 3;  
    }else if(valor > 3000){  
        return 4;  
    }else if(valor > 2000){  
        return 5;  
    }else if(valor > 1500){  
        return 6;  
    }else if(valor > 1000){  
        return 7;  
    }else if(valor > 750){  
        return 8;  
    }else if(valor > 500){  
        return 9;  
    }else if(valor > 100){  
        return 10;  
    }else{  
        return 11;  
    }  
}
```



Método: analisaPraguicidas

Complexidade: 11

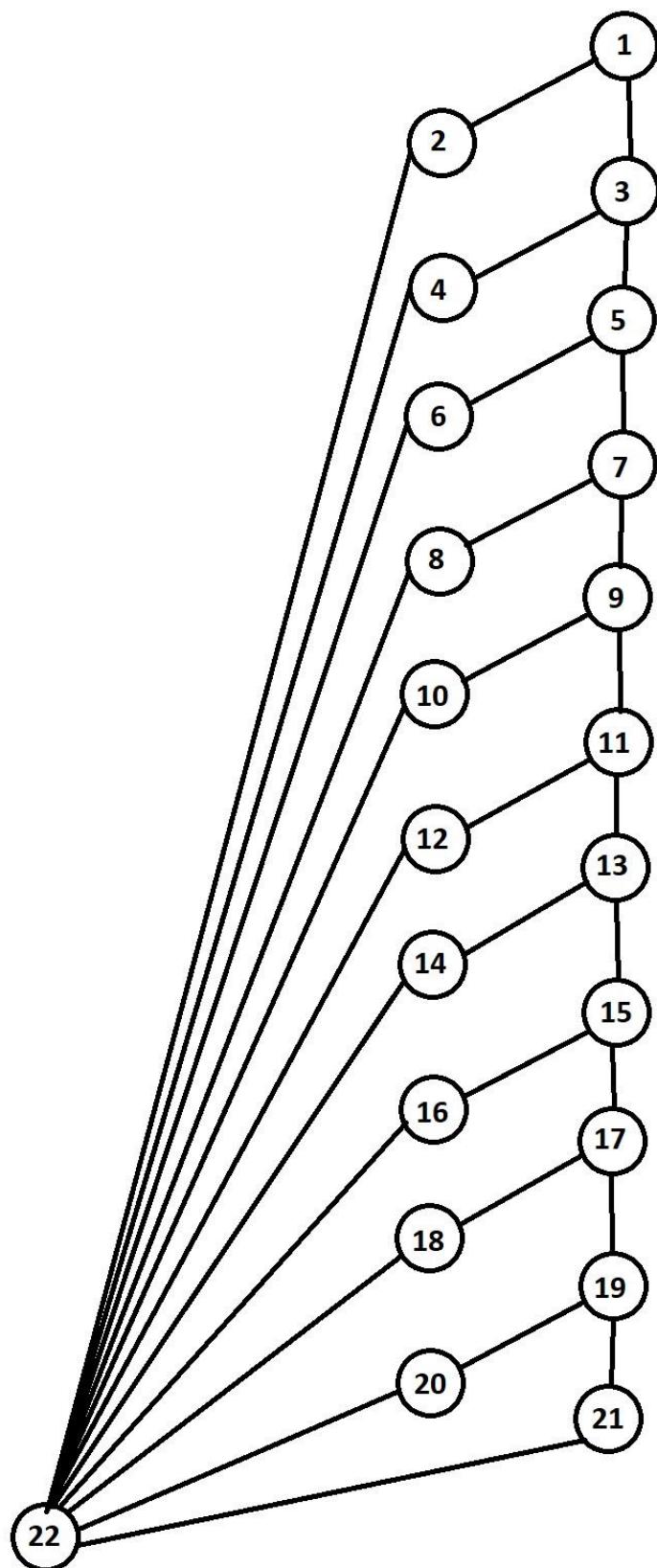
```
public Integer analisaPraguicidas(double valor){  
  
    if(valor > 2){  
        return 1;  
    }else if(valor > 1){  
        return 2;  
    }else if(valor > 0.4){  
        return 3;  
    }else if(valor > 0.2){  
        return 4;  
    }else if(valor > 0.1){  
        return 5;  
    }else if(valor > 0.05){  
        return 6;  
    }else if(valor > 0.025){  
        return 7;  
    }else if(valor > 0.01){  
        return 8;  
    }else if(valor > 0.005){  
        return 9;  
    }else if(valor > 0){  
        return 10;  
    }else{  
        return 11;  
    }  
}
```



Método: analisaGraxasAzeites

Complexidade: 11

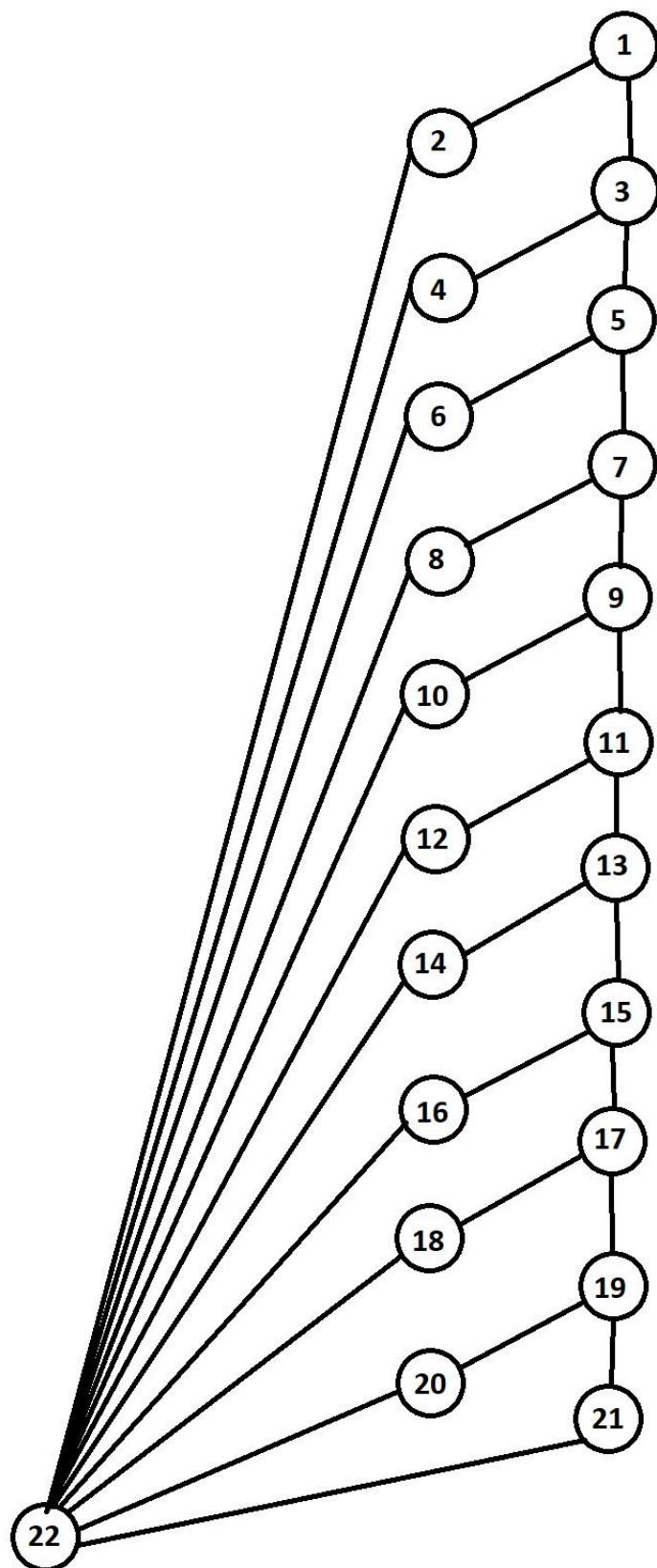
```
public Integer analisaGraxasAzeites(double valor){  
  
    if(valor > 3){  
        return 1;  
    }else if(valor > 2){  
        return 2;  
    }else if(valor > 1){  
        return 3;  
    }else if(valor > 0.6){  
        return 4;  
    }else if(valor > 0.3){  
        return 5;  
    }else if(valor > 0.15){  
        return 6;  
    }else if(valor > 0.08){  
        return 7;  
    }else if(valor > 0.04){  
        return 8;  
    }else if(valor > 0.02){  
        return 9;  
    }else if(valor > 0){  
        return 10;  
    }else{  
        return 11;  
    }  
}
```



Método: analisaSulfatos

Complexidade: 11

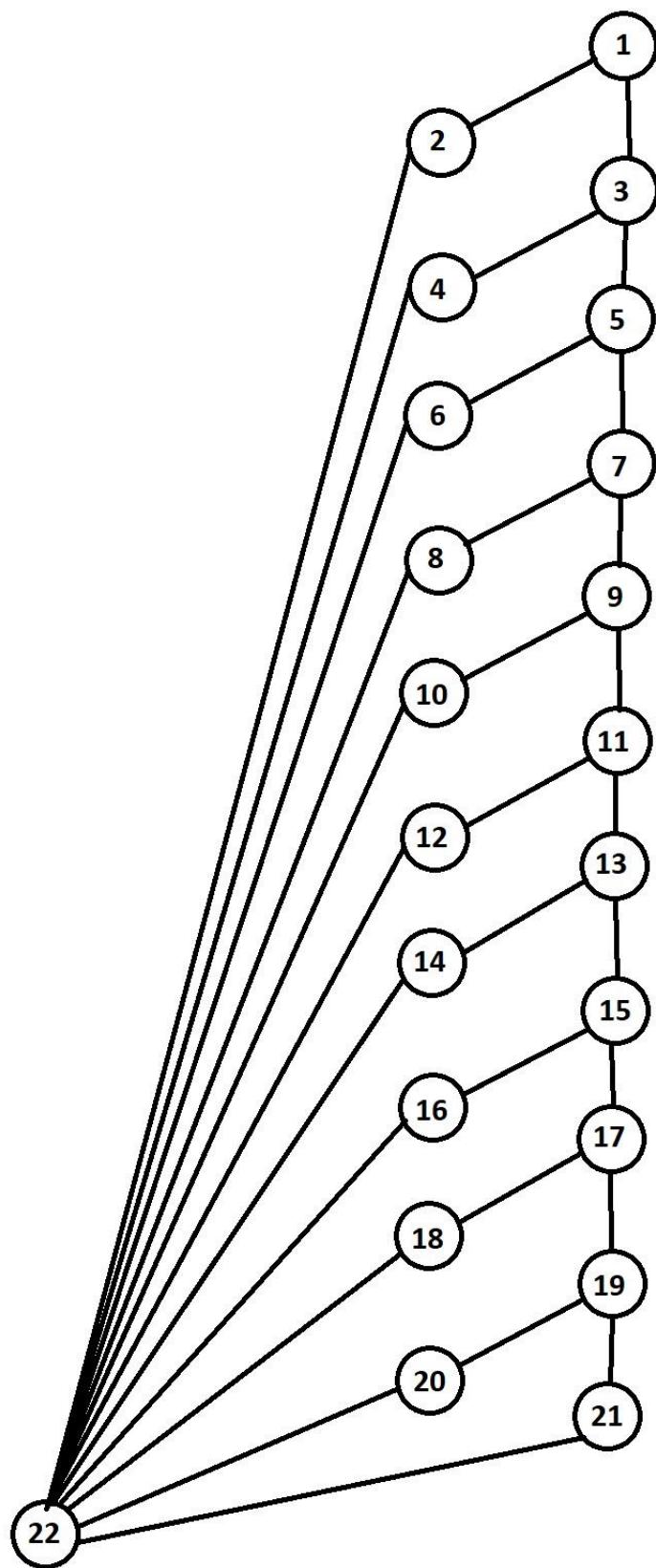
```
public Integer analisaSulfatos(double valor){  
  
    if(valor > 1500){  
        return 1;  
    }else if(valor > 1000){  
        return 2;  
    }else if(valor > 600){  
        return 3;  
    }else if(valor > 400){  
        return 4;  
    }else if(valor > 250){  
        return 5;  
    }else if(valor > 150){  
        return 6;  
    }else if(valor > 100){  
        return 7;  
    }else if(valor > 75){  
        return 8;  
    }else if(valor > 50){  
        return 9;  
    }else if(valor > 0){  
        return 10;  
    }else{  
        return 11;  
    }  
}
```



Método: analisaNitratos

Complexidade: 11

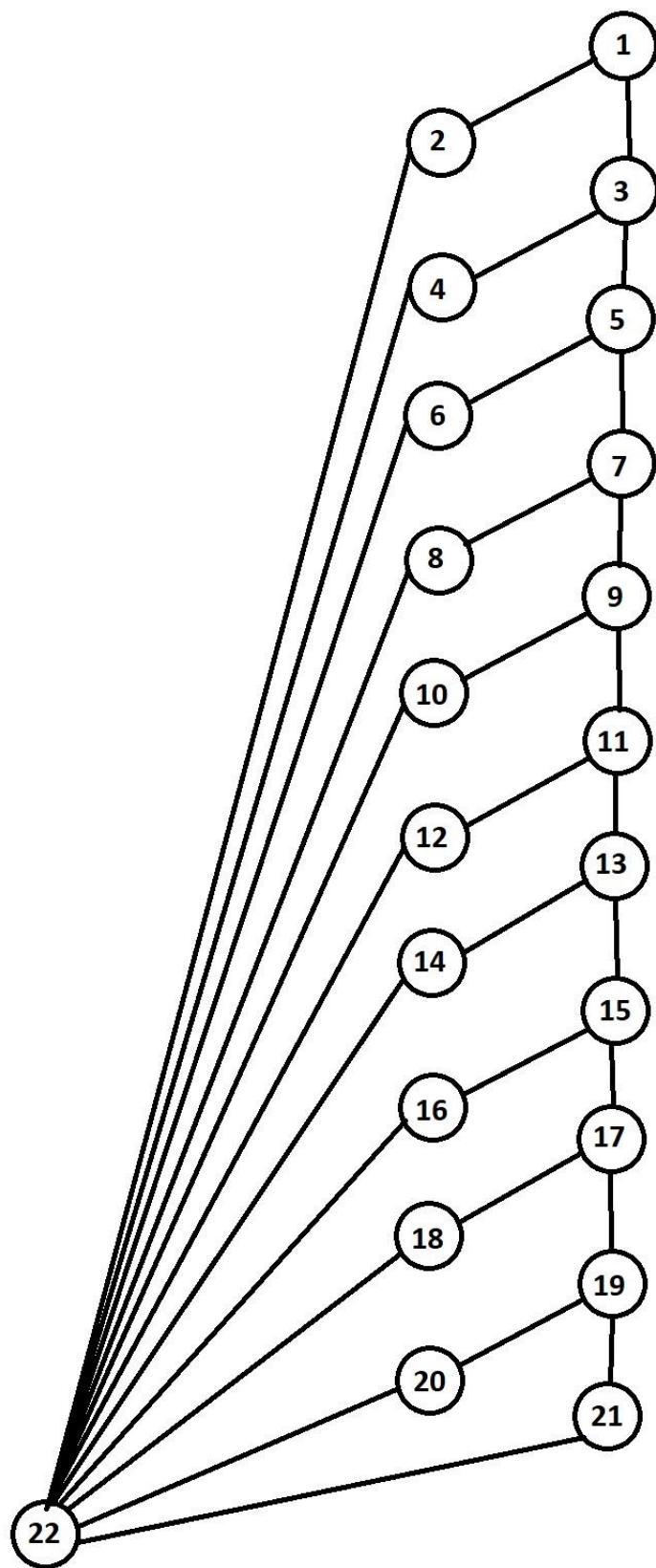
```
public Integer analisaNitratos(double valor){  
  
    if(valor > 100){  
        return 1;  
    }else if(valor > 50){  
        return 2;  
    }else if(valor > 20){  
        return 3;  
    }else if(valor > 15){  
        return 4;  
    }else if(valor > 10){  
        return 5;  
    }else if(valor > 8){  
        return 6;  
    }else if(valor > 6){  
        return 7;  
    }else if(valor > 4){  
        return 8;  
    }else if(valor > 2){  
        return 9;  
    }else if(valor > 0){  
        return 10;  
    }else{  
        return 11;  
    }  
}
```



Método: analisaCianetos

Complexidade: 11

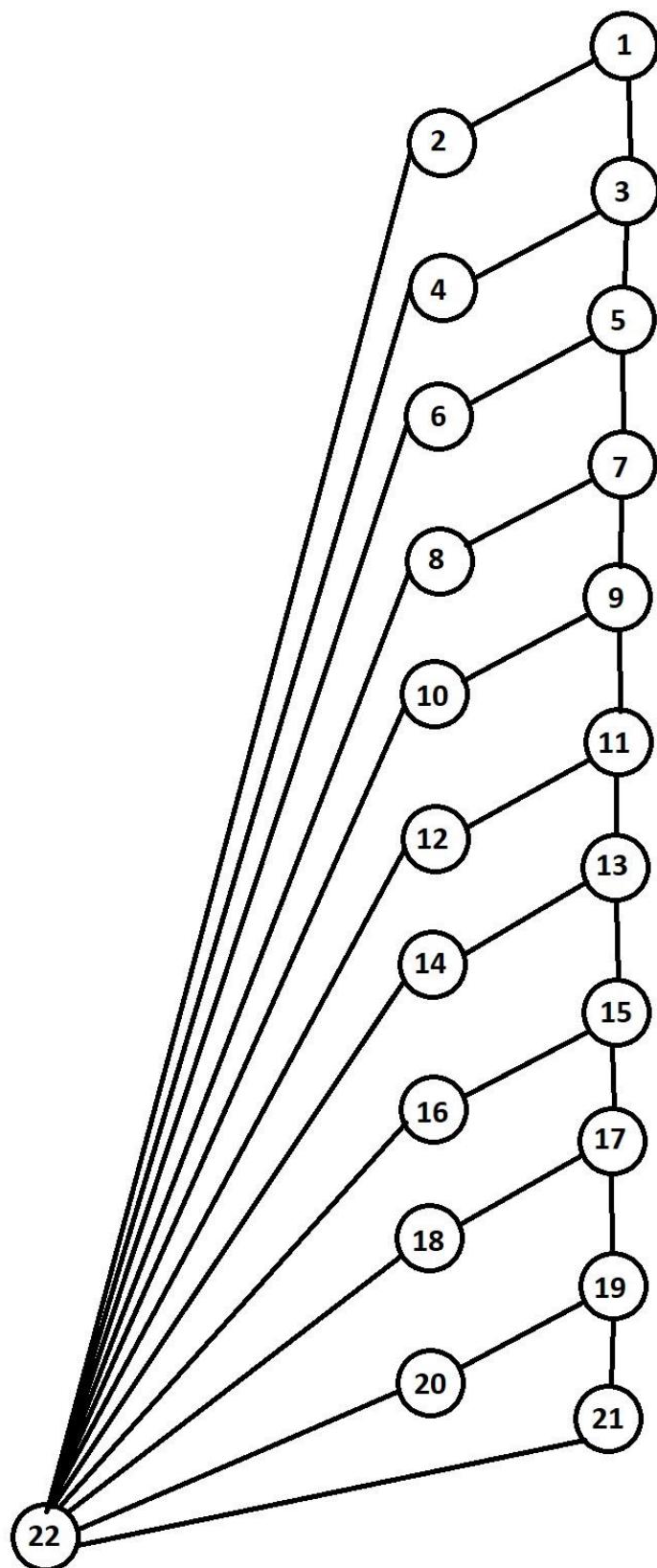
```
public Integer analisaCianetos(double valor){  
  
    if(valor > 1){  
        return 1;  
    }else if(valor > 0.6){  
        return 2;  
    }else if(valor > 0.5){  
        return 3;  
    }else if(valor > 0.4){  
        return 4;  
    }else if(valor > 0.3){  
        return 5;  
    }else if(valor > 0.2){  
        return 6;  
    }else if(valor > 0.1){  
        return 7;  
    }else if(valor > 0.05){  
        return 8;  
    }else if(valor > 0.02){  
        return 9;  
    }else if(valor > 0){  
        return 10;  
    }else{  
        return 11;  
    }  
}
```



Método: analisaCoLivre

Complexidade: 11

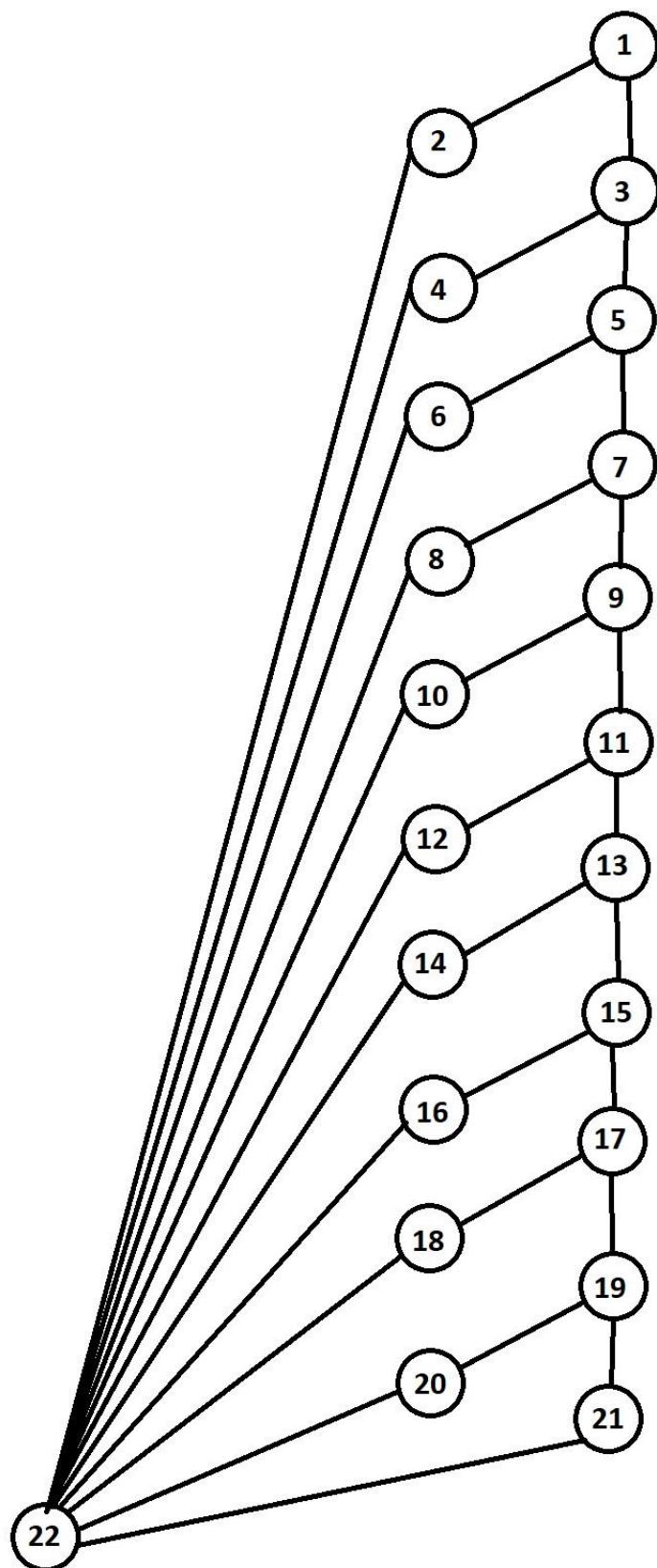
```
public Integer analisaCoLivre(double valor){  
  
    if(valor > 60){  
        return 1;  
    }else if(valor > 50){  
        return 2;  
    }else if(valor > 40){  
        return 3;  
    }else if(valor > 30){  
        return 4;  
    }else if(valor > 20){  
        return 5;  
    }else if(valor > 10){  
        return 6;  
    }else if(valor > 9){  
        return 7;  
    }else if(valor > 8){  
        return 8;  
    }else if(valor > 7){  
        return 9;  
    }else if(valor > 3){  
        return 10;  
    }else{  
        return 11;  
    }  
}
```



Método: analisaMagnesio

Complexidade: 11

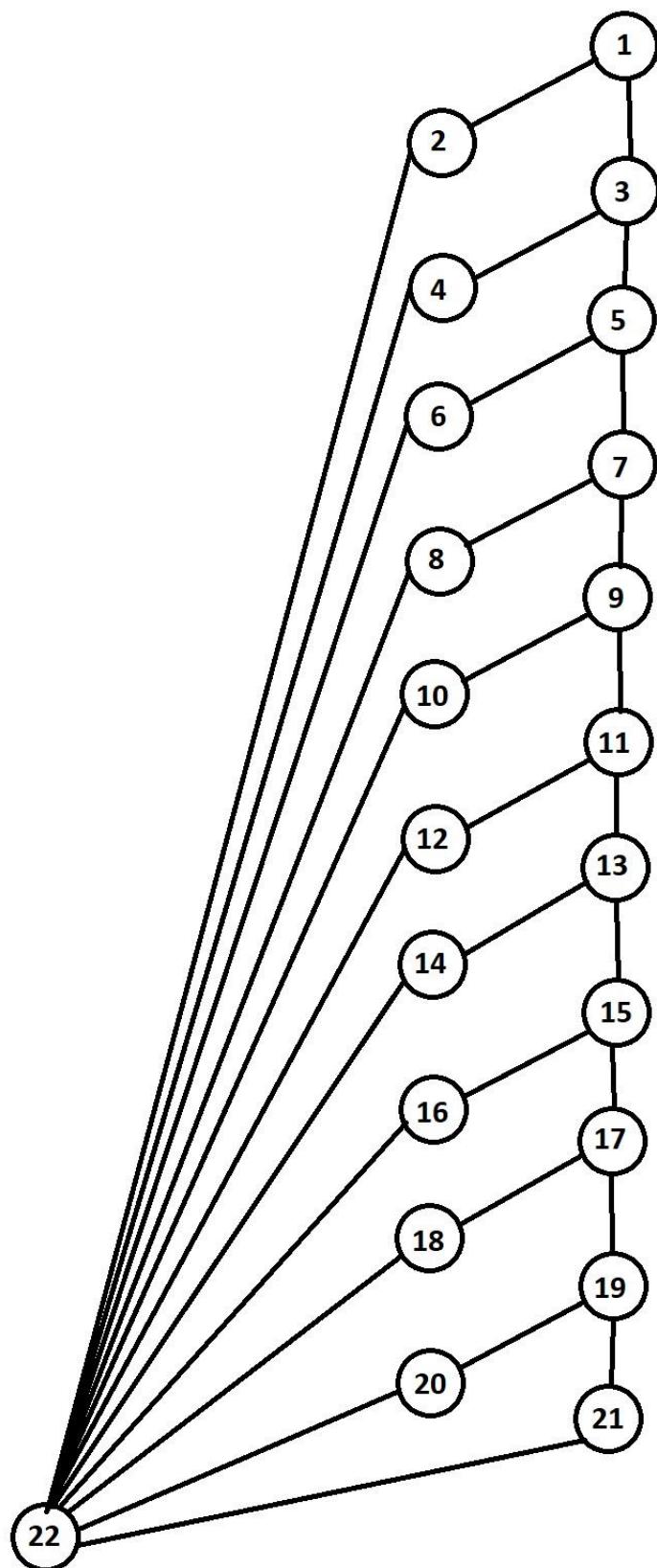
```
public Integer analisaMagnesio(double valor){  
  
    if(valor > 500){  
        return 1;  
    }else if(valor > 300){  
        return 2;  
    }else if(valor > 250){  
        return 3;  
    }else if(valor > 200){  
        return 4;  
    }else if(valor > 150){  
        return 5;  
    }else if(valor > 100){  
        return 6;  
    }else if(valor > 75){  
        return 7;  
    }else if(valor > 50){  
        return 8;  
    }else if(valor > 25){  
        return 9;  
    }else if(valor > 10){  
        return 10;  
    }else{  
        return 11;  
    }  
}
```



Método: analisaFosfatos

Complexidade: 11

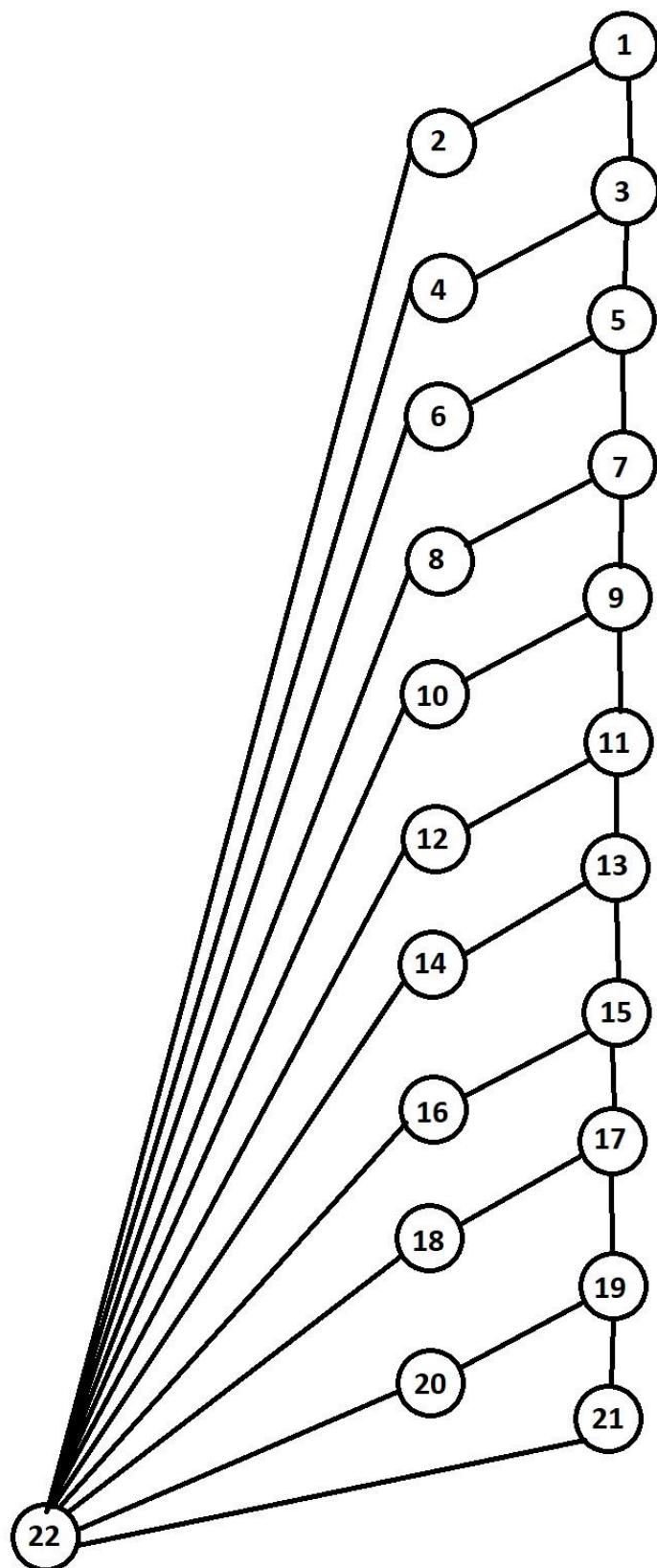
```
public Integer analisaFosfatos(double valor){  
  
    if(valor > 500){  
        return 1;  
    }else if(valor > 300){  
        return 2;  
    }else if(valor > 200){  
        return 3;  
    }else if(valor > 100){  
        return 4;  
    }else if(valor > 50){  
        return 5;  
    }else if(valor > 30){  
        return 6;  
    }else if(valor > 20){  
        return 7;  
    }else if(valor > 10){  
        return 8;  
    }else if(valor > 5){  
        return 9;  
    }else if(valor > 0){  
        return 10;  
    }else{  
        return 11;  
    }  
}
```



Método: analisaNitritos

Complexidade: 11

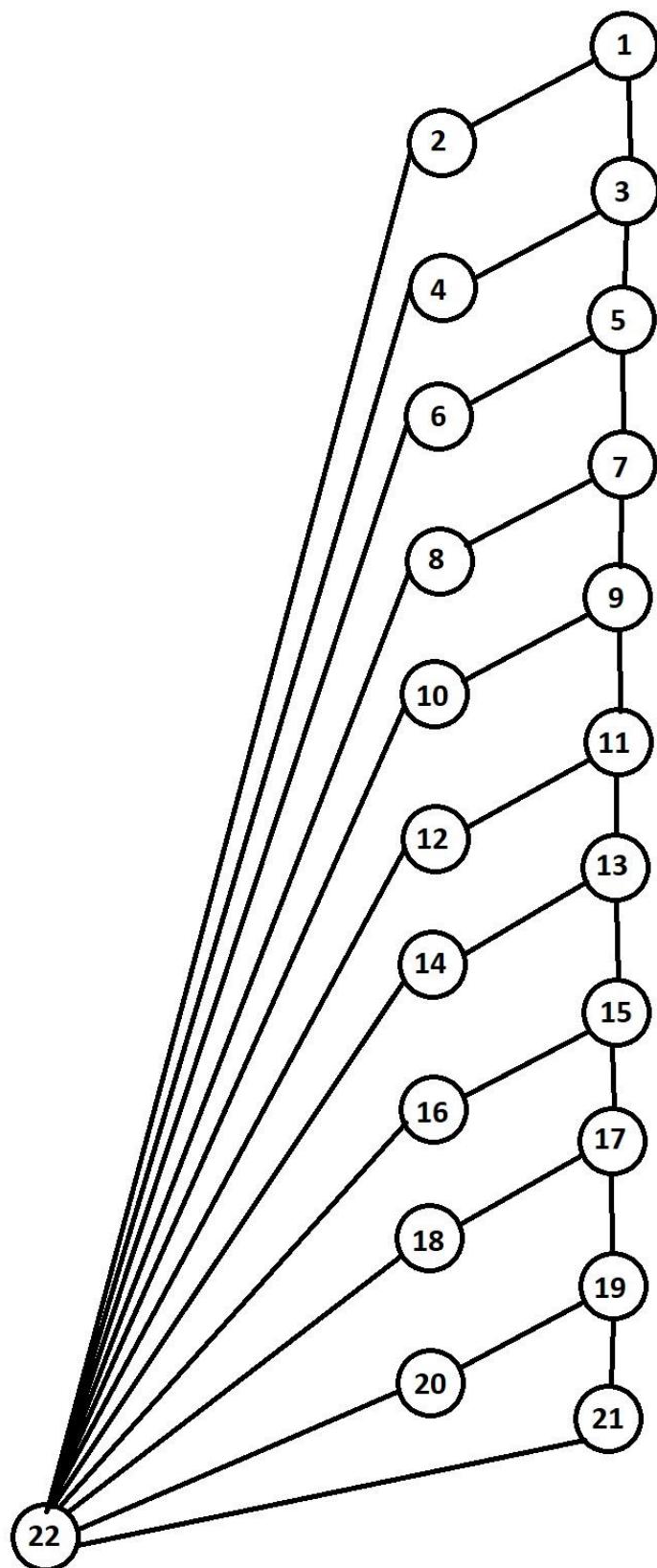
```
public Integer analisaNitritos(double valor){  
  
    if(valor > 1){  
        return 1;  
    }else if(valor > 0.5){  
        return 2;  
    }else if(valor > 0.25){  
        return 3;  
    }else if(valor > 0.20){  
        return 4;  
    }else if(valor > 0.15){  
        return 5;  
    }else if(valor > 0.10){  
        return 6;  
    }else if(valor > 0.05){  
        return 7;  
    }else if(valor > 0.025){  
        return 8;  
    }else if(valor > 0.010){  
        return 9;  
    }else if(valor > 0){  
        return 10;  
    }else{  
        return 11;  
    }  
}
```



Método: analisaDbo

Complexidade: 11

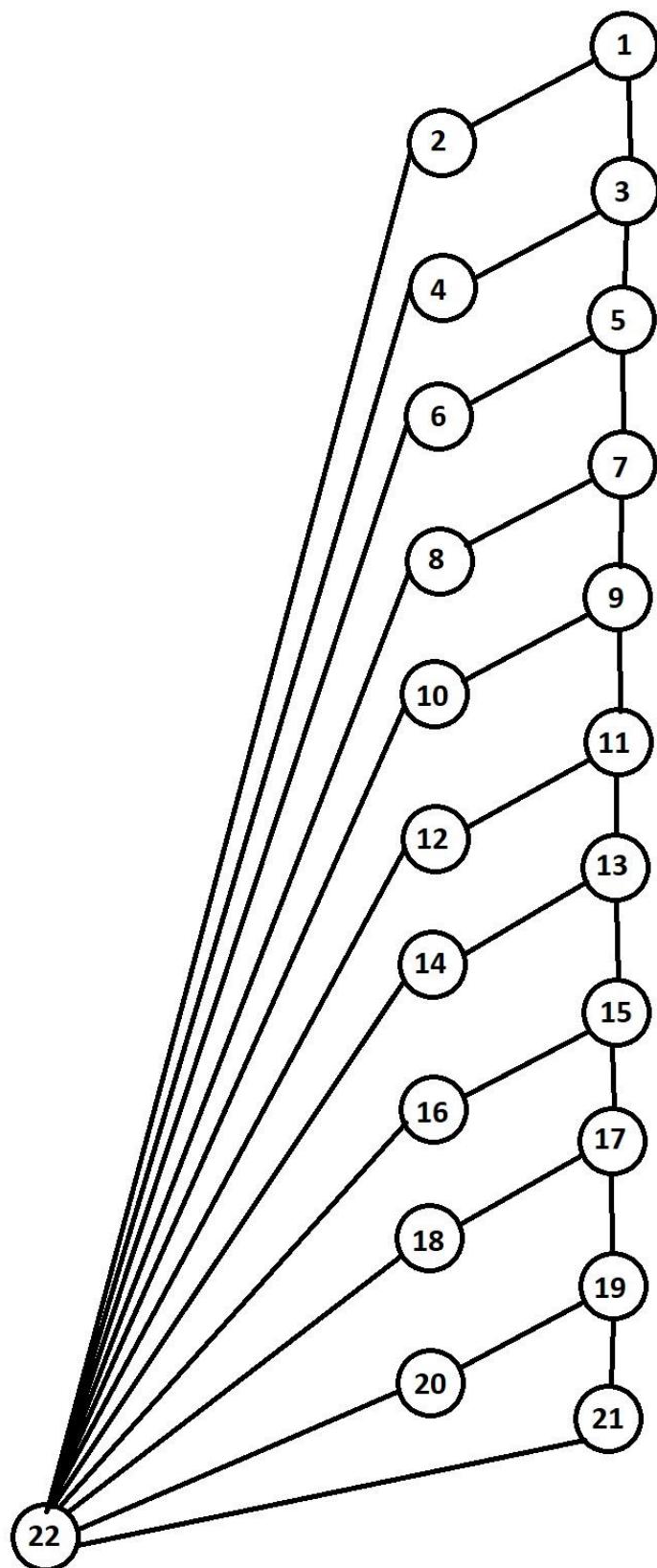
```
public Integer analisaDbo(double valor){  
  
    if(valor > 15){  
        return 1;  
    }else if(valor > 12){  
        return 2;  
    }else if(valor > 10){  
        return 3;  
    }else if(valor > 8){  
        return 4;  
    }else if(valor > 6){  
        return 5;  
    }else if(valor > 5){  
        return 6;  
    }else if(valor > 4){  
        return 7;  
    }else if(valor > 2){  
        return 8;  
    }else if(valor > 1){  
        return 9;  
    }else if(valor > 0.5){  
        return 10;  
    }else{  
        return 11;  
    }  
}
```



Método: analisaCor

Complexidade: 11

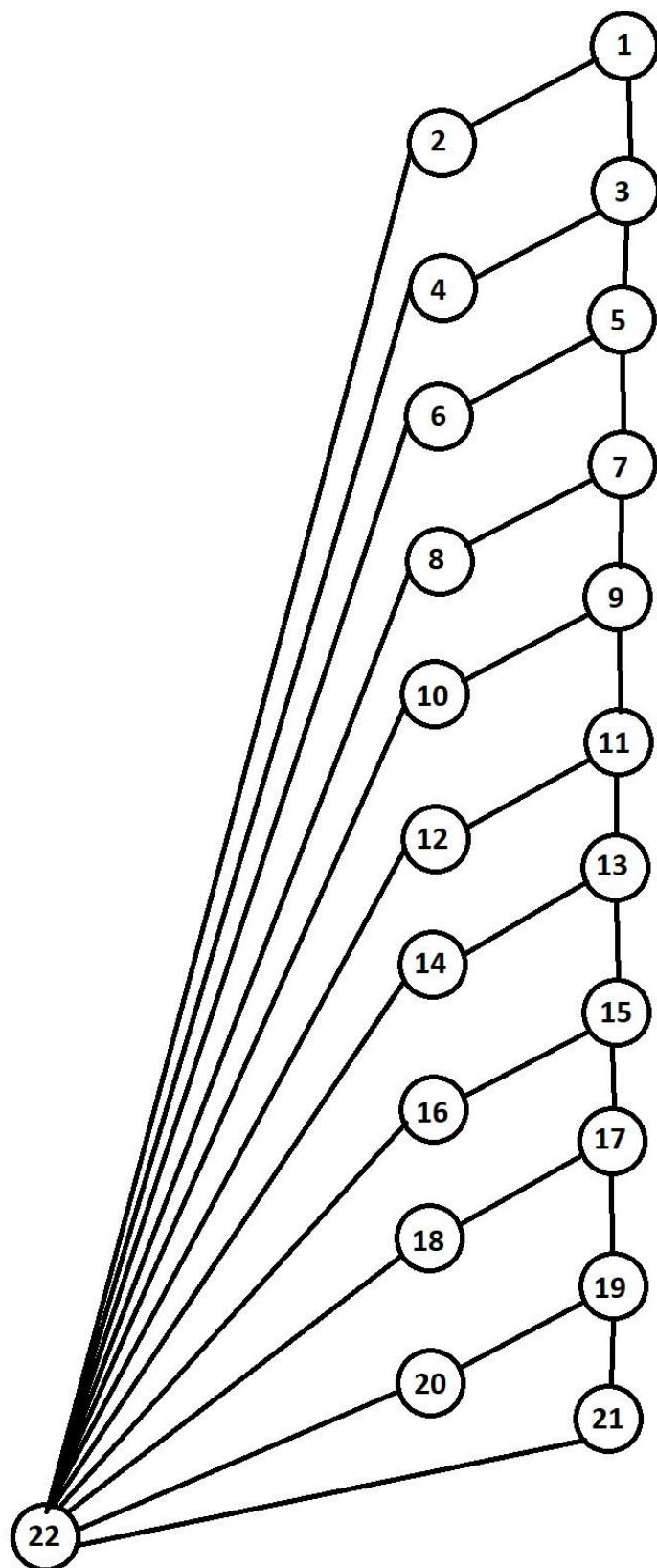
```
public Integer analisaCor(double valor){  
  
    if(valor > 250){  
        return 1;  
    }else if(valor > 100){  
        return 2;  
    }else if(valor > 60){  
        return 3;  
    }else if(valor > 40){  
        return 4;  
    }else if(valor > 30){  
        return 5;  
    }else if(valor > 20){  
        return 6;  
    }else if(valor > 15){  
        return 7;  
    }else if(valor > 10){  
        return 8;  
    }else if(valor > 5){  
        return 9;  
    }else if(valor > 3){  
        return 10;  
    }else{  
        return 11;  
    }  
}
```



Método: analisaTurbidez

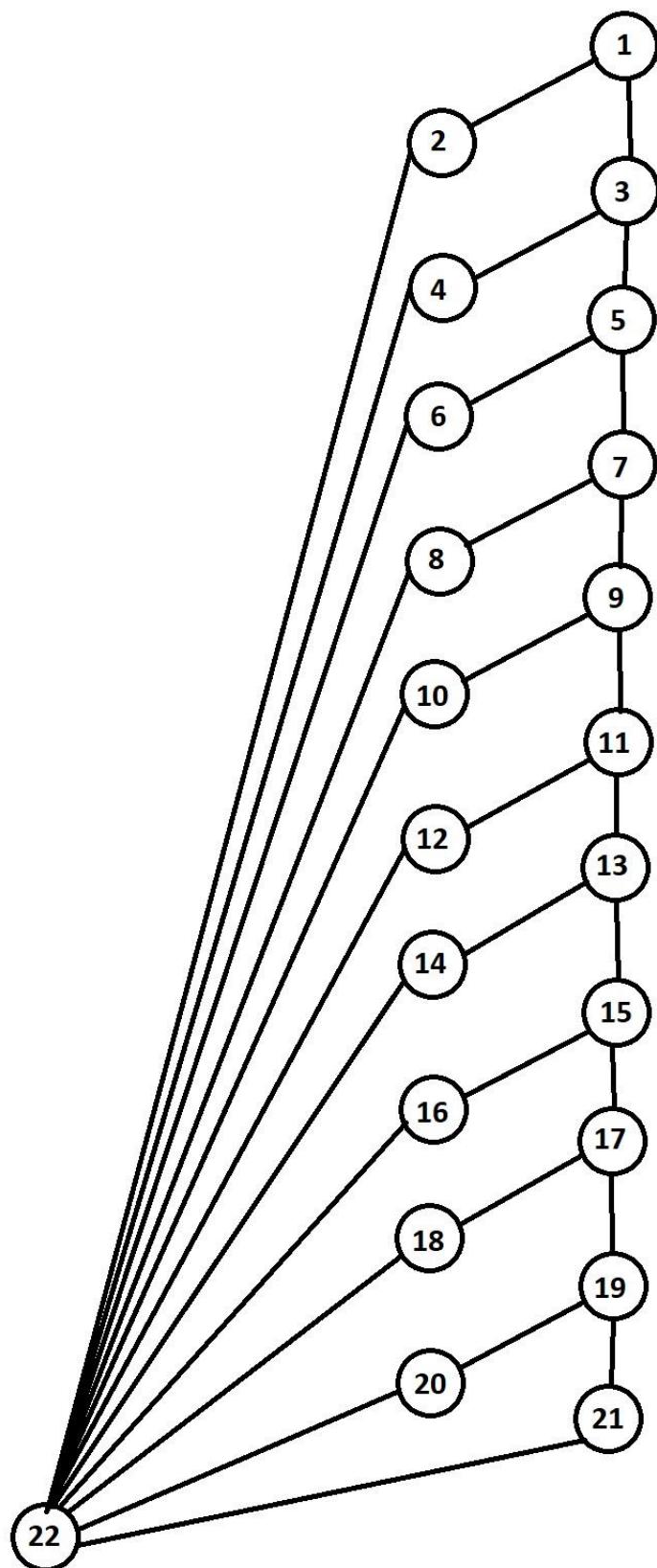
Complexidade: 11

```
public Integer analisaTurbidez(double valor){  
  
    if(valor > 400){  
        return 1;  
    }else if(valor > 250){  
        return 2;  
    }else if(valor > 180){  
        return 3;  
    }else if(valor > 100){  
        return 4;  
    }else if(valor > 50){  
        return 5;  
    }else if(valor > 20){  
        return 6;  
    }else if(valor > 15){  
        return 7;  
    }else if(valor > 10){  
        return 8;  
    }else if(valor > 5){  
        return 9;  
    }else if(valor > 3){  
        return 10;  
    }else{  
        return 11;  
    }  
}
```



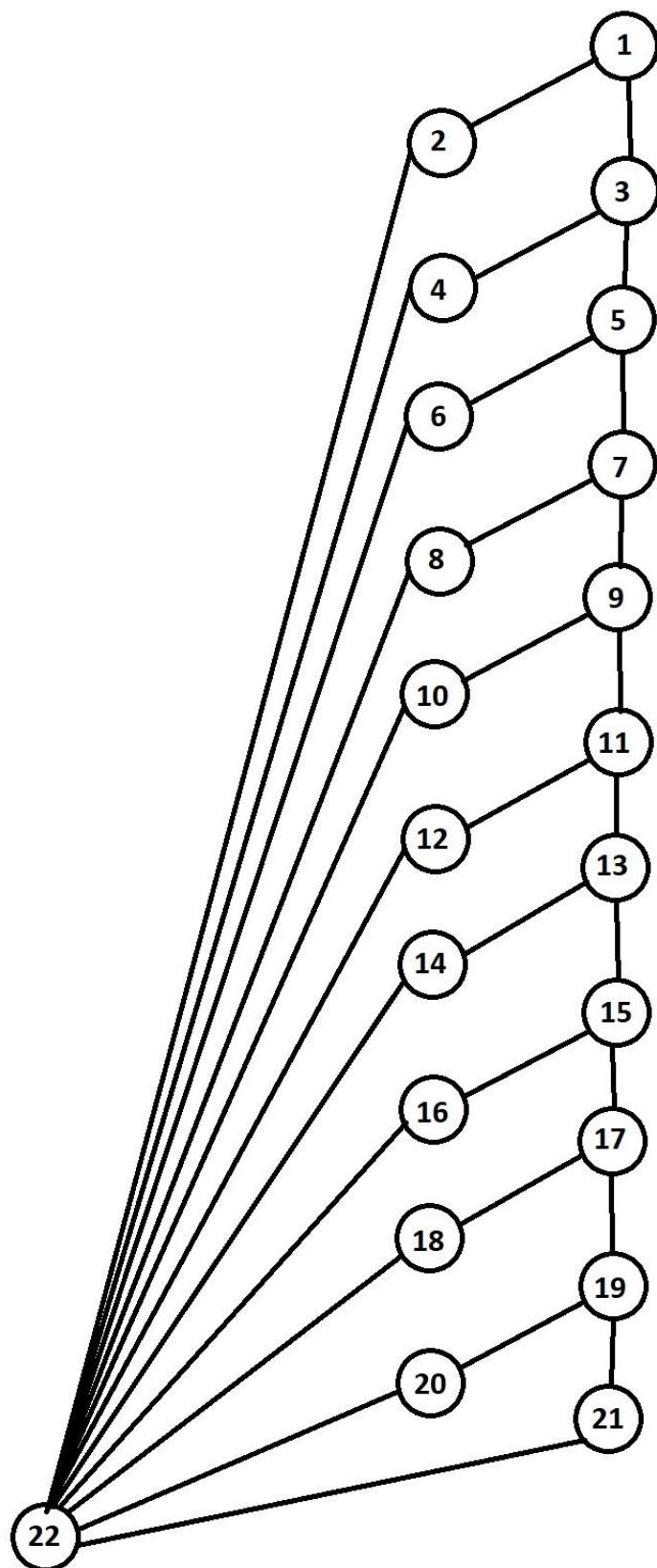
Método: analisaSodio **Complexidade:** 11

```
public Integer analisaSodio(double valor){  
  
    if(valor > 500){  
        return 1;  
    }else if(valor > 300){  
        return 2;  
    }else if(valor > 250){  
        return 3;  
    }else if(valor > 200){  
        return 4;  
    }else if(valor > 150){  
        return 5;  
    }else if(valor > 100){  
        return 6;  
    }else if(valor > 75){  
        return 7;  
    }else if(valor > 50){  
        return 8;  
    }else if(valor > 25){  
        return 9;  
    }else if(valor > 10){  
        return 10;  
    }else{  
        return 11;  
    }  
}
```



Método: analisaCalcio **Complexidade:** 11

```
public Integer analisaCalcio(double valor){  
  
    if(valor > 1000){  
        return 1;  
    }else if(valor > 600){  
        return 2;  
    }else if(valor > 500){  
        return 3;  
    }else if(valor > 400){  
        return 4;  
    }else if(valor > 300){  
        return 5;  
    }else if(valor > 200){  
        return 6;  
    }else if(valor > 150){  
        return 7;  
    }else if(valor > 100){  
        return 8;  
    }else if(valor > 50){  
        return 9;  
    }else if(valor > 10){  
        return 10;  
    }else{  
        return 11;  
    }  
}
```



4.2 – Comparação do resultado com Source Code Metrics

Através da ferramenta Source Code Metrics pode ser verificado a Complexidade Ciclomática dos métodos. Segue abaixo o resultado obtido para os métodos das classes de domínio através da coluna VG. Pode ser comparado com os valores encontrados através do grafo.

Class	Method	VG
ProfissionalAnalise	analiseAmostra	24
ProfissionalAnalise	analisaCalcio	11
ProfissionalAnalise	analisaCianetos	11
ProfissionalAnalise	analisaCloreto	11
ProfissionalAnalise	analisaCoLivre	11
ProfissionalAnalise	analisaColiformesTotais	11
ProfissionalAnalise	analisaCondutividade	11
ProfissionalAnalise	analisaCor	11
ProfissionalAnalise	analisaDbo	11
ProfissionalAnalise	analisaDetergentes	11
ProfissionalAnalise	analisaDurezaAlcalinidade	11
ProfissionalAnalise	analisaFosfatos	11
ProfissionalAnalise	analisaGraxasAzeites	11
ProfissionalAnalise	analisaMagnesio	11
ProfissionalAnalise	analisaNitratos	11
ProfissionalAnalise	analisaNitritos	11
ProfissionalAnalise	analisaNitrogenioAmoniacal	11
ProfissionalAnalise	analisaOxigenioDissolvido	11
ProfissionalAnalise	analisaPh	11
ProfissionalAnalise	analisaPraguicidas	11
ProfissionalAnalise	analisaReducaoPermanganato	11
ProfissionalAnalise	analisaSodio	11
ProfissionalAnalise	analisaSolidosDissolvidos	11
ProfissionalAnalise	analisaSulfatos	11
ProfissionalAnalise	analisaTurbidez	11
Contrato	getContratoEstadoCodigo	6
Contrato	setContratoEstadoCodigo	6
Contrato	setContratoEstadoName	6
Contrato	equals	5
ProfissionalEspecialista	calculalqa	5
Amostra	buscarAmostrasAnalise	4
Contrato	buscarContratoColeta	3
Contrato	buscarContratoFechamento	2
ContratoEstadoFactory	create	2
Amostra	Amostra	1
Amostra	dropAmostra	1
Amostra	getCodigo	1
Amostra	getContrato	1
Amostra	getLocal	1
Amostra	getProfissionalColeta	1
Amostra	obterAmostra	1

Amostra	obterAmostras	1
Amostra	saveAmostra	1
Amostra	setCodigo	1
Amostra	setContrato	1
Amostra	setLocal	1
Amostra	setParameter	1
Amostra	setProfissionalColeta	1
Amostra	updateAmostra	1
Analise	Analise	1
Analise	Analise	1
Analise	getAmostra	1
Analise	getCodigo	1
Analise	getProfissionalAnalise	1
Analise	getResultado	1
Analise	getServico	1
Analise	getValor	1
Analise	setAmostra	1
Analise	setCodigo	1
Analise	setProfissionalAnalise	1
Analise	setResultado	1
Analise	setServico	1
Analise	setValor	1
Cliente	Cliente	1
Cliente	Cliente	1
Cliente	dropCliente	1
Cliente	getCelular	1
Cliente	getCodigo	1
Cliente	getEmail	1
Cliente	getIdentificacao	1
Cliente	getNome	1
Cliente	getTelefone	1
Cliente	obterCliente	1
Cliente	obterClientes	1
Cliente	saveCliente	1
Cliente	setCelular	1
Cliente	setCodigo	1
Cliente	setEmail	1
Cliente	setIdentificacao	1
Cliente	setNome	1
Cliente	setParameter	1
Cliente	setTelefone	1
Cliente	updateCliente	1
Contrato	Contrato	1
Contrato	Contrato	1
Contrato	dropContrato	1
Contrato	getCliente	1
Contrato	getCodigo	1
Contrato	getContratoEstado	1
Contrato	getDescricao	1
Contrato	getEmpresa	1
Contrato	getLocal	1
Contrato	hashCode	1

Contrato	obterContrato	1
Contrato	obterContratos	1
Contrato	saveContrato	1
Contrato	setCliente	1
Contrato	setCodigo	1
Contrato	setContratoEstado	1
Contrato	setDescricao	1
Contrato	setEmpresa	1
Contrato	setLocal	1
Contrato	setParameter	1
Contrato	updateContrato	1
Contrato	updateContrato	1
ContratoEstado	aberto	1
ContratoEstado	analiseFinalizada	1
ContratoEstado	coletaFinalizada	1
ContratoEstado	emAnalise	1
ContratoEstado	emColeta	1
ContratoEstado	fechado	1
ContratoEstado	getEstado	1
ContratoEstadoAberto	aberto	1
ContratoEstadoAberto	analiseFinalizada	1
ContratoEstadoAberto	coletaFinalizada	1
ContratoEstadoAberto	emAnalise	1
ContratoEstadoAberto	emColeta	1
ContratoEstadoAberto	fechado	1
ContratoEstadoAberto	getEstado	1
ContratoEstadoAnaliseFinalizada	aberto	1
ContratoEstadoAnaliseFinalizada	analiseFinalizada	1
ContratoEstadoAnaliseFinalizada	coletaFinalizada	1
ContratoEstadoAnaliseFinalizada	emAnalise	1
ContratoEstadoAnaliseFinalizada	emColeta	1
ContratoEstadoAnaliseFinalizada	fechado	1
ContratoEstadoAnaliseFinalizada	getEstado	1
ContratoEstadoColetaFinalizada	aberto	1
ContratoEstadoColetaFinalizada	analiseFinalizada	1
ContratoEstadoColetaFinalizada	coletaFinalizada	1
ContratoEstadoColetaFinalizada	emAnalise	1
ContratoEstadoColetaFinalizada	emColeta	1
ContratoEstadoColetaFinalizada	fechado	1
ContratoEstadoColetaFinalizada	getEstado	1
ContratoEstadoEmAnalise	aberto	1
ContratoEstadoEmAnalise	analiseFinalizada	1
ContratoEstadoEmAnalise	coletaFinalizada	1
ContratoEstadoEmAnalise	emAnalise	1
ContratoEstadoEmAnalise	emColeta	1
ContratoEstadoEmAnalise	fechado	1
ContratoEstadoEmAnalise	getEstado	1
ContratoEstadoEmColeta	aberto	1
ContratoEstadoEmColeta	analiseFinalizada	1
ContratoEstadoEmColeta	coletaFinalizada	1
ContratoEstadoEmColeta	emAnalise	1
ContratoEstadoEmColeta	emColeta	1

ContratoEstadoEmColeta	fechado	1
ContratoEstadoEmColeta	getEstado	1
ContratoEstadoFechado	aberto	1
ContratoEstadoFechado	analiseFinalizada	1
ContratoEstadoFechado	coletaFinalizada	1
ContratoEstadoFechado	emAnalise	1
ContratoEstadoFechado	emColeta	1
ContratoEstadoFechado	fechado	1
ContratoEstadoFechado	getEstado	1
ContratoServico	ContratoServico	1
ContratoServico	ContratoServico	1
ContratoServico	getCodigo	1
ContratoServico	getContrato	1
ContratoServico	getServico	1
ContratoServico	setCodigo	1
ContratoServico	setContrato	1
ContratoServico	setServico	1
Empresa	Empresa	1
Empresa	Empresa	1
Empresa	dropEmpresa	1
Empresa	getBairro	1
Empresa	getCidade	1
Empresa	getCodigo	1
Empresa	getComplemento	1
Empresa	getEstado	1
Empresa	getIdentificacao	1
Empresa	getLogradouro	1
Empresa	getNome	1
Empresa	getNumero	1
Empresa	getPais	1
Empresa	obterEmpresa	1
Empresa	obterEmpresas	1
Empresa	saveEmpresa	1
Empresa	setBairro	1
Empresa	setCidade	1
Empresa	setCodigo	1
Empresa	setComplemento	1
Empresa	setEstado	1
Empresa	setIdentificacao	1
Empresa	setLogradouro	1
Empresa	setNome	1
Empresa	setNumero	1
Empresa	setPais	1
Empresa	setParameter	1
Empresa	updateEmpresa	1
Funcionario	Funcionario	1
Funcionario	Funcionario	1
Funcionario	getCargo	1
Funcionario	getCodigo	1
Funcionario	getEmpresa	1
Funcionario	getIdentificacao	1
Funcionario	getNome	1

Funcionario	setCargo	1
Funcionario	setCodigo	1
Funcionario	setEmpresa	1
Funcionario	setIdentificacao	1
Funcionario	setNome	1
Local	Local	1
Local	Local	1
Local	dropLocal	1
Local	getBairro	1
Local	getCidade	1
Local	getCodigo	1
Local	getComplemento	1
Local	getEstado	1
Local	getLogradouro	1
Local	getNumero	1
Local	obterLocal	1
Local	saveLocal	1
Local	setBairro	1
Local	setCidade	1
Local	setCodigo	1
Local	setComplemento	1
Local	setEstado	1
Local	setLogradouro	1
Local	setNumero	1
Local	setParameter	1
ProfissionalAnalise	obterFuncionario	1
ProfissionalColeta	coletaAmostra	1
ProfissionalEspecialista	obterFuncionario	1
Resultado	Resultado	1
Resultado	Resultado	1
Resultado	getCodigo	1
Resultado	getNome	1
Resultado	setCodigo	1
Resultado	setNome	1
Servico	Servico	1
Servico	Servico	1
Servico	dropServico	1
Servico	getCodigo	1
Servico	getNome	1
Servico	obterServico	1
Servico	obterServicos	1
Servico	saveServico	1
Servico	setCodigo	1
Servico	setNome	1
Servico	setParameter	1
Servico	updateServico	1

5 - Análise do valor limite e teste unitário

Após descrevemos os métodos, precisamos determinar os valores para teste de cada um deles e o valor esperado no final. Segue abaixo a análise dos métodos de maior complexidade.

Método: analisaPh

Valor	Resultado
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	10
10.1	11

```
public void testAnalisaPhRetorna1(){
    assertEquals(new Integer("1"), profissionalAnalise.analisaPh(1));
}
public void testAnalisaPhRetorna2(){
    assertEquals(new Integer("2"), profissionalAnalise.analisaPh(2));
}
public void testAnalisaPhRetorna3(){
    assertEquals(new Integer("3"), profissionalAnalise.analisaPh(3));
}
public void testAnalisaPhRetorna4(){
    assertEquals(new Integer("4"), profissionalAnalise.analisaPh(4));
}
public void testAnalisaPhRetorna5(){
    assertEquals(new Integer("5"), profissionalAnalise.analisaPh(5));
}
public void testAnalisaPhRetorna6(){
    assertEquals(new Integer("6"), profissionalAnalise.analisaPh(6));
}
public void testAnalisaPhRetorna7(){
    assertEquals(new Integer("7"), profissionalAnalise.analisaPh(7));
}
public void testAnalisaPhRetorna8(){
    assertEquals(new Integer("8"), profissionalAnalise.analisaPh(8));
}
public void testAnalisaPhRetorna9(){
    assertEquals(new Integer("9"), profissionalAnalise.analisaPh(9));
}
public void testAnalisaPhRetorna10(){
    assertEquals(new Integer("10"), profissionalAnalise.analisaPh(10));
}
public void testAnalisaPhRetorna11(){
    assertEquals(new Integer("11"), profissionalAnalise.analisaPh(10.1));
}
```

Método: analisaCondutovidade

Valor	Resultado
16001	1
12001	2
8001	3
5001	4
3001	5
2501	6
2001	7
1501	8
1251	9
750	10
749	11

```
public void testAnalisaCondutovidadeRetorna1() {
    assertEquals(new Integer("1"), profissionalAnalise.analisaCondutovidade(16001));
}
public void testAnalisaCondutovidadeRetorna2() {
    assertEquals(new Integer("2"), profissionalAnalise.analisaCondutovidade(12001));
}
public void testAnalisaCondutovidadeRetorna3() {
    assertEquals(new Integer("3"), profissionalAnalise.analisaCondutovidade(8001));
}
public void testAnalisaCondutovidadeRetorna4() {
    assertEquals(new Integer("4"), profissionalAnalise.analisaCondutovidade(5001));
}
public void testAnalisaCondutovidadeRetorna5() {
    assertEquals(new Integer("5"), profissionalAnalise.analisaCondutovidade(3001));
}
public void testAnalisaCondutovidadeRetorna6() {
    assertEquals(new Integer("6"), profissionalAnalise.analisaCondutovidade(2501));
}
public void testAnalisaCondutovidadeRetorna7() {
    assertEquals(new Integer("7"), profissionalAnalise.analisaCondutovidade(2001));
}
public void testAnalisaCondutovidadeRetorna8() {
    assertEquals(new Integer("8"), profissionalAnalise.analisaCondutovidade(1501));
}
public void testAnalisaCondutovidadeRetorna9() {
    assertEquals(new Integer("9"), profissionalAnalise.analisaCondutovidade(1251));
}
public void testAnalisaCondutovidadeRetorna10() {
    assertEquals(new Integer("10"), profissionalAnalise.analisaCondutovidade(750));
}
public void testAnalisaCondutovidadeRetorna11() {
    assertEquals(new Integer("11"), profissionalAnalise.analisaCondutovidade(749));
}
```

Método: analisaOxigenioDissolvido

Valor	Resultado
0.9	1
1.9	2
2.9	3
3.4	4
3.9	5
4.9	6
5.9	7
6.4	8
6.9	9
7.4	10
7.5	11

```
public void testAnalisaOxigenioDissolvidoRetorna1() {
    assertEquals(new Integer("1"), profissionalAnalise.analisaOxigenioDissolvido(0.9));
}
public void testAnalisaOxigenioDissolvidoRetorna2() {
    assertEquals(new Integer("2"), profissionalAnalise.analisaOxigenioDissolvido(1.9));
}
public void testAnalisaOxigenioDissolvidoRetorna3() {
    assertEquals(new Integer("3"), profissionalAnalise.analisaOxigenioDissolvido(2.9));
}
public void testAnalisaOxigenioDissolvidoRetorna4() {
    assertEquals(new Integer("4"), profissionalAnalise.analisaOxigenioDissolvido(3.4));
}
public void testAnalisaOxigenioDissolvidoRetorna5() {
    assertEquals(new Integer("5"), profissionalAnalise.analisaOxigenioDissolvido(3.9));
}
public void testAnalisaOxigenioDissolvidoRetorna6() {
    assertEquals(new Integer("6"), profissionalAnalise.analisaOxigenioDissolvido(4.9));
}
public void testAnalisaOxigenioDissolvidoRetorna7() {
    assertEquals(new Integer("7"), profissionalAnalise.analisaOxigenioDissolvido(5.9));
}
public void testAnalisaOxigenioDissolvidoRetorna8() {
    assertEquals(new Integer("8"), profissionalAnalise.analisaOxigenioDissolvido(6.4));
}
public void testAnalisaOxigenioDissolvidoRetorna9() {
    assertEquals(new Integer("9"), profissionalAnalise.analisaOxigenioDissolvido(6.9));
}
public void testAnalisaOxigenioDissolvidoRetorna10() {
    assertEquals(new Integer("10"), profissionalAnalise.analisaOxigenioDissolvido(7.4));
}
public void testAnalisaOxigenioDissolvidoRetorna11() {
    assertEquals(new Integer("11"), profissionalAnalise.analisaOxigenioDissolvido(7.5));
}
```

Método: analisa ReducaoPermanganato

Valor	Resultado
16	1
13	2
11	3
9	4
7	5
6	6
5	7
4	8
3	9
0.6	10
0.5	11

```
public void testAnalisaReducaoPermanganatoRetorna1(){
    assertEquals(new Integer("1"), profissionalAnalise.analisaReducaoPermanganato(16));
}
public void testAnalisaReducaoPermanganatoRetorna2(){
    assertEquals(new Integer("2"), profissionalAnalise.analisaReducaoPermanganato(13));
}
public void testAnalisaReducaoPermanganatoRetorna3(){
    assertEquals(new Integer("3"), profissionalAnalise.analisaReducaoPermanganato(11));
}
public void testAnalisaReducaoPermanganatoRetorna4(){
    assertEquals(new Integer("4"), profissionalAnalise.analisaReducaoPermanganato(9));
}
public void testAnalisaReducaoPermanganatoRetorna5(){
    assertEquals(new Integer("5"), profissionalAnalise.analisaReducaoPermanganato(7));
}
public void testAnalisaReducaoPermanganatoRetorna6(){
    assertEquals(new Integer("6"), profissionalAnalise.analisaReducaoPermanganato(6));
}
public void testAnalisaReducaoPermanganatoRetorna7(){
    assertEquals(new Integer("7"), profissionalAnalise.analisaReducaoPermanganato(5));
}
public void testAnalisaReducaoPermanganatoRetorna8(){
    assertEquals(new Integer("8"), profissionalAnalise.analisaReducaoPermanganato(4));
}
public void testAnalisaReducaoPermanganatoRetorna9(){
    assertEquals(new Integer("9"), profissionalAnalise.analisaReducaoPermanganato(3));
}
public void testAnalisaReducaoPermanganatoRetorna10(){
    assertEquals(new Integer("10"), profissionalAnalise.analisaReducaoPermanganato(0.6));
}
public void testAnalisaReducaoPermanganatoRetorna11(){
    assertEquals(new Integer("11"), profissionalAnalise.analisaReducaoPermanganato(0.5));
}
```

Método: analisaColiformesTotais

Valor	Resultado
14001	1
10001	2
7001	3
5001	4
4001	5
3001	6
2001	7
1501	8
1001	9
51	10
50	11

```
public void testAnalisaColiformesTotaisRetorna1(){
    assertEquals(new Integer("1"), profissionalAnalise.analisaColiformesTotais(14001));
}
public void testAnalisaColiformesTotaisRetorna2(){
    assertEquals(new Integer("2"), profissionalAnalise.analisaColiformesTotais(10001));
}
public void testAnalisaColiformesTotaisRetorna3(){
    assertEquals(new Integer("3"), profissionalAnalise.analisaColiformesTotais(7001));
}
public void testAnalisaColiformesTotaisRetorna4(){
    assertEquals(new Integer("4"), profissionalAnalise.analisaColiformesTotais(5001));
}
public void testAnalisaColiformesTotaisRetorna5(){
    assertEquals(new Integer("5"), profissionalAnalise.analisaColiformesTotais(4001));
}
public void testAnalisaColiformesTotaisRetorna6(){
    assertEquals(new Integer("6"), profissionalAnalise.analisaColiformesTotais(3001));
}
public void testAnalisaColiformesTotaisRetorna7(){
    assertEquals(new Integer("7"), profissionalAnalise.analisaColiformesTotais(2001));
}
public void testAnalisaColiformesTotaisRetorna8(){
    assertEquals(new Integer("8"), profissionalAnalise.analisaColiformesTotais(1501));
}
public void testAnalisaColiformesTotaisRetorna9(){
    assertEquals(new Integer("9"), profissionalAnalise.analisaColiformesTotais(1001));
}
public void testAnalisaColiformesTotaisRetorna10(){
    assertEquals(new Integer("10"), profissionalAnalise.analisaColiformesTotais(51));
}
public void testAnalisaColiformesTotaisRetorna11(){
    assertEquals(new Integer("11"), profissionalAnalise.analisaColiformesTotais(50));
}
```

Método: analisaNitrogenioAmoniacal

Valor	Resultado
1.26	1
1.01	2
0.76	3
0.51	4
0.41	5
0.31	6
0.21	7
0.11	8
0.06	9
0.01	10
0	11

```
public void testAnalisaNitrogenioAmoniacalRetorna1(){
    assertEquals(new Integer("1"), profissionalAnalise.analisaNitrogenioAmoniacal(1.26));
}
public void testAnalisaNitrogenioAmoniacalRetorna2(){
    assertEquals(new Integer("2"), profissionalAnalise.analisaNitrogenioAmoniacal(1.01));
}
public void testAnalisaNitrogenioAmoniacalRetorna3(){
    assertEquals(new Integer("3"), profissionalAnalise.analisaNitrogenioAmoniacal(0.76));
}
public void testAnalisaNitrogenioAmoniacalRetorna4(){
    assertEquals(new Integer("4"), profissionalAnalise.analisaNitrogenioAmoniacal(0.51));
}
public void testAnalisaNitrogenioAmoniacalRetorna5(){
    assertEquals(new Integer("5"), profissionalAnalise.analisaNitrogenioAmoniacal(0.41));
}
public void testAnalisaNitrogenioAmoniacalRetorna6(){
    assertEquals(new Integer("6"), profissionalAnalise.analisaNitrogenioAmoniacal(0.31));
}
public void testAnalisaNitrogenioAmoniacalRetorna7(){
    assertEquals(new Integer("7"), profissionalAnalise.analisaNitrogenioAmoniacal(0.21));
}
public void testAnalisaNitrogenioAmoniacalRetorna8(){
    assertEquals(new Integer("8"), profissionalAnalise.analisaNitrogenioAmoniacal(0.11));
}
public void testAnalisaNitrogenioAmoniacalRetorna9(){
    assertEquals(new Integer("9"), profissionalAnalise.analisaNitrogenioAmoniacal(0.06));
}
public void testAnalisaNitrogenioAmoniacalRetorna10(){
    assertEquals(new Integer("10"), profissionalAnalise.analisaNitrogenioAmoniacal(0.01));
}
public void testAnalisaNitrogenioAmoniacalRetorna11(){
    assertEquals(new Integer("11"), profissionalAnalise.analisaNitrogenioAmoniacal(0));
}
```

Método: analisaCloreto

Valor	Resultado
1501	1
1001	2
701	3
501	4
301	5
201	6
151	7
101	8
51	9
1	10
0	11

```
public void testAnalisaCloretoRetorna1(){
    assertEquals(new Integer("1"), profissionalAnalise.analisaCloreto(1501));
}
public void testAnalisaCloretoRetorna2(){
    assertEquals(new Integer("2"), profissionalAnalise.analisaCloreto(1001));
}
public void testAnalisaCloretoRetorna3(){
    assertEquals(new Integer("3"), profissionalAnalise.analisaCloreto(701));
}
public void testAnalisaCloretoRetorna4(){
    assertEquals(new Integer("4"), profissionalAnalise.analisaCloreto(501));
}
public void testAnalisaCloretoRetorna5(){
    assertEquals(new Integer("5"), profissionalAnalise.analisaCloreto(301));
}
public void testAnalisaCloretoRetorna6(){
    assertEquals(new Integer("6"), profissionalAnalise.analisaCloreto(201));
}
public void testAnalisaCloretoRetorna7(){
    assertEquals(new Integer("7"), profissionalAnalise.analisaCloreto(151));
}
public void testAnalisaCloretoRetorna8(){
    assertEquals(new Integer("8"), profissionalAnalise.analisaCloreto(101));
}
public void testAnalisaCloretoRetorna9(){
    assertEquals(new Integer("9"), profissionalAnalise.analisaCloreto(51));
}
public void testAnalisaCloretoRetorna10(){
    assertEquals(new Integer("10"), profissionalAnalise.analisaCloreto(1));
}
public void testAnalisaCloretoRetorna11(){
    assertEquals(new Integer("11"), profissionalAnalise.analisaCloreto(0));
}
```

Método: analisaDetergentes

Valor	Resultado
3.1	1
2.1	2
1.6	3
1.1	4
0.76	5
0.51	6
0.26	7
0.11	8
0.07	9
0.01	10
0	11

```
public void testAnalisaDetergentesRetorna1(){
    assertEquals(new Integer("1"), profissionalAnalise.analisaDetergentes(3.1));
}
public void testAnalisaDetergentesRetorna2(){
    assertEquals(new Integer("2"), profissionalAnalise.analisaDetergentes(2.1));
}
public void testAnalisaDetergentesRetorna3(){
    assertEquals(new Integer("3"), profissionalAnalise.analisaDetergentes(1.6));
}
public void testAnalisaDetergentesRetorna4(){
    assertEquals(new Integer("4"), profissionalAnalise.analisaDetergentes(1.1));
}
public void testAnalisaDetergentesRetorna5(){
    assertEquals(new Integer("5"), profissionalAnalise.analisaDetergentes(0.76));
}
public void testAnalisaDetergentesRetorna6(){
    assertEquals(new Integer("6"), profissionalAnalise.analisaDetergentes(0.51));
}
public void testAnalisaDetergentesRetorna7(){
    assertEquals(new Integer("7"), profissionalAnalise.analisaDetergentes(0.26));
}
public void testAnalisaDetergentesRetorna8(){
    assertEquals(new Integer("8"), profissionalAnalise.analisaDetergentes(0.11));
}
public void testAnalisaDetergentesRetorna9(){
    assertEquals(new Integer("9"), profissionalAnalise.analisaDetergentes(0.07));
}
public void testAnalisaDetergentesRetorna10(){
    assertEquals(new Integer("10"), profissionalAnalise.analisaDetergentes(0.01));
}
public void testAnalisaDetergentesRetorna11(){
    assertEquals(new Integer("11"), profissionalAnalise.analisaDetergentes(0));
}
```

Método: analisaDurezaAlcalinidade

Valor	Resultado
1501	1
1001	2
801	3
601	4
501	5
401	6
301	7
201	8
101	9
26	10
25	11

```
public void testAnalisaDurezaAlcalinidadeRetorna1(){
    assertEquals(new Integer("1"), profissionalAnalise.analisaDurezaAlcalinidade(1501));
}
public void testAnalisaDurezaAlcalinidadeRetorna2(){
    assertEquals(new Integer("2"), profissionalAnalise.analisaDurezaAlcalinidade(1001));
}
public void testAnalisaDurezaAlcalinidadeRetorna3(){
    assertEquals(new Integer("3"), profissionalAnalise.analisaDurezaAlcalinidade(801));
}
public void testAnalisaDurezaAlcalinidadeRetorna4(){
    assertEquals(new Integer("4"), profissionalAnalise.analisaDurezaAlcalinidade(601));
}
public void testAnalisaDurezaAlcalinidadeRetorna5(){
    assertEquals(new Integer("5"), profissionalAnalise.analisaDurezaAlcalinidade(501));
}
public void testAnalisaDurezaAlcalinidadeRetorna6(){
    assertEquals(new Integer("6"), profissionalAnalise.analisaDurezaAlcalinidade(401));
}
public void testAnalisaDurezaAlcalinidadeRetorna7(){
    assertEquals(new Integer("7"), profissionalAnalise.analisaDurezaAlcalinidade(301));
}
public void testAnalisaDurezaAlcalinidadeRetorna8(){
    assertEquals(new Integer("8"), profissionalAnalise.analisaDurezaAlcalinidade(201));
}
public void testAnalisaDurezaAlcalinidadeRetorna9(){
    assertEquals(new Integer("9"), profissionalAnalise.analisaDurezaAlcalinidade(101));
}
public void testAnalisaDurezaAlcalinidadeRetorna10(){
    assertEquals(new Integer("10"), profissionalAnalise.analisaDurezaAlcalinidade(26));
}
public void testAnalisaDurezaAlcalinidadeRetorna11(){
    assertEquals(new Integer("11"), profissionalAnalise.analisaDurezaAlcalinidade(25));
}
```

Método: analisaSolidosDissolvidos

Valor	Resultado
20001	1
10001	2
5001	3
3001	4
2001	5
1501	6
1001	7
751	8
501	9
101	10
100	11

```
public void testAnalisaSolidosDissolvidosRetorna1(){
    assertEquals(new Integer("1"), profissionalAnalise.analisaSolidosDissolvidos(20001));
}
public void testAnalisaSolidosDissolvidosRetorna2(){
    assertEquals(new Integer("2"), profissionalAnalise.analisaSolidosDissolvidos(10001));
}
public void testAnalisaSolidosDissolvidosRetorna3(){
    assertEquals(new Integer("3"), profissionalAnalise.analisaSolidosDissolvidos(5001));
}
public void testAnalisaSolidosDissolvidosRetorna4(){
    assertEquals(new Integer("4"), profissionalAnalise.analisaSolidosDissolvidos(3001));
}
public void testAnalisaSolidosDissolvidosRetorna5(){
    assertEquals(new Integer("5"), profissionalAnalise.analisaSolidosDissolvidos(2001));
}
public void testAnalisaSolidosDissolvidosRetorna6(){
    assertEquals(new Integer("6"), profissionalAnalise.analisaSolidosDissolvidos(1501));
}
public void testAnalisaSolidosDissolvidosRetorna7(){
    assertEquals(new Integer("7"), profissionalAnalise.analisaSolidosDissolvidos(1001));
}
public void testAnalisaSolidosDissolvidosRetorna8(){
    assertEquals(new Integer("8"), profissionalAnalise.analisaSolidosDissolvidos(751));
}
public void testAnalisaSolidosDissolvidosRetorna9(){
    assertEquals(new Integer("9"), profissionalAnalise.analisaSolidosDissolvidos(501));
}
public void testAnalisaSolidosDissolvidosRetorna10(){
    assertEquals(new Integer("10"), profissionalAnalise.analisaSolidosDissolvidos(101));
}
public void testAnalisaSolidosDissolvidosRetorna11(){
    assertEquals(new Integer("11"), profissionalAnalise.analisaSolidosDissolvidos(100));
}
```

Método: analisaPraguicidas

Valor	Resultado
2.01	1
1.01	2
0.41	3
0.21	4
0.11	5
0.06	6
0.026	7
0.011	8
0.006	9
0.001	10
0	11

```
public void testAnalisaPraguicidasRetorna1(){
    assertEquals(new Integer("1"), profissionalAnalise.analisaPraguicidas(2.01));
}
public void testAnalisaPraguicidasRetorna2(){
    assertEquals(new Integer("2"), profissionalAnalise.analisaPraguicidas(1.01));
}
public void testAnalisaPraguicidasRetorna3(){
    assertEquals(new Integer("3"), profissionalAnalise.analisaPraguicidas(0.41));
}
public void testAnalisaPraguicidasRetorna4(){
    assertEquals(new Integer("4"), profissionalAnalise.analisaPraguicidas(0.21));
}
public void testAnalisaPraguicidasRetorna5(){
    assertEquals(new Integer("5"), profissionalAnalise.analisaPraguicidas(0.11));
}
public void testAnalisaPraguicidasRetorna6(){
    assertEquals(new Integer("6"), profissionalAnalise.analisaPraguicidas(0.06));
}
public void testAnalisaPraguicidasRetorna7(){
    assertEquals(new Integer("7"), profissionalAnalise.analisaPraguicidas(0.026));
}
public void testAnalisaPraguicidasRetorna8(){
    assertEquals(new Integer("8"), profissionalAnalise.analisaPraguicidas(0.011));
}
public void testAnalisaPraguicidasRetorna9(){
    assertEquals(new Integer("9"), profissionalAnalise.analisaPraguicidas(0.006));
}
public void testAnalisaPraguicidasRetorna10(){
    assertEquals(new Integer("10"), profissionalAnalise.analisaPraguicidas(0.001));
}
public void testAnalisaPraguicidasRetorna11(){
    assertEquals(new Integer("11"), profissionalAnalise.analisaPraguicidas(0));
}
```

Método: analisaGraxasAzeites

Valor	Resultado
3.1	1
2.1	2
1.1	3
0.7	4
0.4	5
0.16	6
0.09	7
0.05	8
0.03	9
0.01	10
0	11

```
public void testAnalisaGraxasAzeitesRetorna1(){
    assertEquals(new Integer("1"), profissionalAnalise.analisaGraxasAzeites(3.1));
}
public void testAnalisaGraxasAzeitesRetorna2(){
    assertEquals(new Integer("2"), profissionalAnalise.analisaGraxasAzeites(2.1));
}
public void testAnalisaGraxasAzeitesRetorna3(){
    assertEquals(new Integer("3"), profissionalAnalise.analisaGraxasAzeites(1.1));
}
public void testAnalisaGraxasAzeitesRetorna4(){
    assertEquals(new Integer("4"), profissionalAnalise.analisaGraxasAzeites(0.7));
}
public void testAnalisaGraxasAzeitesRetorna5(){
    assertEquals(new Integer("5"), profissionalAnalise.analisaGraxasAzeites(0.4));
}
public void testAnalisaGraxasAzeitesRetorna6(){
    assertEquals(new Integer("6"), profissionalAnalise.analisaGraxasAzeites(0.16));
}
public void testAnalisaGraxasAzeitesRetorna7(){
    assertEquals(new Integer("7"), profissionalAnalise.analisaGraxasAzeites(0.09));
}
public void testAnalisaGraxasAzeitesRetorna8(){
    assertEquals(new Integer("8"), profissionalAnalise.analisaGraxasAzeites(0.05));
}
public void testAnalisaGraxasAzeitesRetorna9(){
    assertEquals(new Integer("9"), profissionalAnalise.analisaGraxasAzeites(0.03));
}
public void testAnalisaGraxasAzeitesRetorna10(){
    assertEquals(new Integer("10"), profissionalAnalise.analisaGraxasAzeites(0.01));
}
public void testAnalisaGraxasAzeitesRetorna11(){
    assertEquals(new Integer("11"), profissionalAnalise.analisaGraxasAzeites(0));
}
```

Método: analisaSulfatos

Valor	Resultado
1501	1
1001	2
601	3
401	4
251	5
151	6
101	7
76	8
51	9
1	10
0	11

```
public void testAnalisaSulfatosRetorna1(){
    assertEquals(new Integer("1"), profissionalAnalise.analisaSulfatos(1501));
}
public void testAnalisaSulfatosRetorna2(){
    assertEquals(new Integer("2"), profissionalAnalise.analisaSulfatos(1001));
}
public void testAnalisaSulfatosRetorna3(){
    assertEquals(new Integer("3"), profissionalAnalise.analisaSulfatos(601));
}
public void testAnalisaSulfatosRetorna4(){
    assertEquals(new Integer("4"), profissionalAnalise.analisaSulfatos(401));
}
public void testAnalisaSulfatosRetorna5(){
    assertEquals(new Integer("5"), profissionalAnalise.analisaSulfatos(251));
}
public void testAnalisaSulfatosRetorna6(){
    assertEquals(new Integer("6"), profissionalAnalise.analisaSulfatos(151));
}
public void testAnalisaSulfatosRetorna7(){
    assertEquals(new Integer("7"), profissionalAnalise.analisaSulfatos(101));
}
public void testAnalisaSulfatosRetorna8(){
    assertEquals(new Integer("8"), profissionalAnalise.analisaSulfatos(76));
}
public void testAnalisaSulfatosRetorna9(){
    assertEquals(new Integer("9"), profissionalAnalise.analisaSulfatos(51));
}
public void testAnalisaSulfatosRetorna10(){
    assertEquals(new Integer("10"), profissionalAnalise.analisaSulfatos(1));
}
public void testAnalisaSulfatosRetorna11(){
    assertEquals(new Integer("11"), profissionalAnalise.analisaSulfatos(0));
}
```

Método: analisaNitratos

Valor	Resultado
101	1
51	2
21	3
16	4
11	5
9	6
7	7
5	8
3	9
1	10
0	11

```
public void testAnalisaNitratosRetorna1(){
    assertEquals(new Integer("1"), profissionalAnalise.analisaNitratos(101));
}
public void testAnalisaNitratosRetorna2(){
    assertEquals(new Integer("2"), profissionalAnalise.analisaNitratos(51));
}
public void testAnalisaNitratosRetorna3(){
    assertEquals(new Integer("3"), profissionalAnalise.analisaNitratos(21));
}
public void testAnalisaNitratosRetorna4(){
    assertEquals(new Integer("4"), profissionalAnalise.analisaNitratos(16));
}
public void testAnalisaNitratosRetorna5(){
    assertEquals(new Integer("5"), profissionalAnalise.analisaNitratos(11));
}
public void testAnalisaNitratosRetorna6(){
    assertEquals(new Integer("6"), profissionalAnalise.analisaNitratos(9));
}
public void testAnalisaNitratosRetorna7(){
    assertEquals(new Integer("7"), profissionalAnalise.analisaNitratos(7));
}
public void testAnalisaNitratosRetorna8(){
    assertEquals(new Integer("8"), profissionalAnalise.analisaNitratos(5));
}
public void testAnalisaNitratosRetorna9(){
    assertEquals(new Integer("9"), profissionalAnalise.analisaNitratos(3));
}
public void testAnalisaNitratosRetorna10(){
    assertEquals(new Integer("10"), profissionalAnalise.analisaNitratos(1));
}
public void testAnalisaNitratosRetorna11(){
    assertEquals(new Integer("11"), profissionalAnalise.analisaNitratos(0));
}
```

Método: analisaCianetos

Valor	Resultado
1.01	1
0.61	2
0.51	3
0.41	4
0.31	5
0.21	6
0.11	7
0.06	8
0.03	9
0.01	10
0	11

```
public void testAnalisaCianetosRetorna1(){
    assertEquals(new Integer("1"), profissionalAnalise.analisaCianetos(1.01));
}
public void testAnalisaCianetosRetorna2(){
    assertEquals(new Integer("2"), profissionalAnalise.analisaCianetos(0.61));
}
public void testAnalisaCianetosRetorna3(){
    assertEquals(new Integer("3"), profissionalAnalise.analisaCianetos(0.51));
}
public void testAnalisaCianetosRetorna4(){
    assertEquals(new Integer("4"), profissionalAnalise.analisaCianetos(0.41));
}
public void testAnalisaCianetosRetorna5(){
    assertEquals(new Integer("5"), profissionalAnalise.analisaCianetos(0.31));
}
public void testAnalisaCianetosRetorna6(){
    assertEquals(new Integer("6"), profissionalAnalise.analisaCianetos(0.21));
}
public void testAnalisaCianetosRetorna7(){
    assertEquals(new Integer("7"), profissionalAnalise.analisaCianetos(0.11));
}
public void testAnalisaCianetosRetorna8(){
    assertEquals(new Integer("8"), profissionalAnalise.analisaCianetos(0.06));
}
public void testAnalisaCianetosRetorna9(){
    assertEquals(new Integer("9"), profissionalAnalise.analisaCianetos(0.03));
}
public void testAnalisaCianetosRetorna10(){
    assertEquals(new Integer("10"), profissionalAnalise.analisaCianetos(0.01));
}
public void testAnalisaCianetosRetorna11(){
    assertEquals(new Integer("11"), profissionalAnalise.analisaCianetos(0));
}
```

Método: analisaCoLivre

Valor	Resultado
61	1
51	2
41	3
31	4
21	5
11	6
10	7
9	8
8	9
4	10
3	11

```
public void testAnalisaCoLivreRetorna1(){
    assertEquals(new Integer("1"), profissionalAnalise.analisaCoLivre(61));
}
public void testAnalisaCoLivreRetorna2(){
    assertEquals(new Integer("2"), profissionalAnalise.analisaCoLivre(51));
}
public void testAnalisaCoLivreRetorna3(){
    assertEquals(new Integer("3"), profissionalAnalise.analisaCoLivre(41));
}
public void testAnalisaCoLivreRetorna4(){
    assertEquals(new Integer("4"), profissionalAnalise.analisaCoLivre(31));
}
public void testAnalisaCoLivreRetorna5(){
    assertEquals(new Integer("5"), profissionalAnalise.analisaCoLivre(21));
}
public void testAnalisaCoLivreRetorna6(){
    assertEquals(new Integer("6"), profissionalAnalise.analisaCoLivre(11));
}
public void testAnalisaCoLivreRetorna7(){
    assertEquals(new Integer("7"), profissionalAnalise.analisaCoLivre(10));
}
public void testAnalisaCoLivreRetorna8(){
    assertEquals(new Integer("8"), profissionalAnalise.analisaCoLivre(9));
}
public void testAnalisaCoLivreRetorna9(){
    assertEquals(new Integer("9"), profissionalAnalise.analisaCoLivre(8));
}
public void testAnalisaCoLivreRetorna10(){
    assertEquals(new Integer("10"), profissionalAnalise.analisaCoLivre(4));
}
public void testAnalisaCoLivreRetorna11(){
    assertEquals(new Integer("11"), profissionalAnalise.analisaCoLivre(3));
}
```

Método: analisaMagnesio

Valor	Resultado
501	1
301	2
251	3
201	4
151	5
101	6
76	7
51	8
26	9
11	10
10	11

```
public void testAnalisaMagnesioRetorna1(){
    assertEquals(new Integer("1"), profissionalAnalise.analisaMagnesio(501));
}
public void testAnalisaMagnesioRetorna2(){
    assertEquals(new Integer("2"), profissionalAnalise.analisaMagnesio(301));
}
public void testAnalisaMagnesioRetorna3(){
    assertEquals(new Integer("3"), profissionalAnalise.analisaMagnesio(251));
}
public void testAnalisaMagnesioRetorna4(){
    assertEquals(new Integer("4"), profissionalAnalise.analisaMagnesio(201));
}
public void testAnalisaMagnesioRetorna5(){
    assertEquals(new Integer("5"), profissionalAnalise.analisaMagnesio(151));
}
public void testAnalisaMagnesioRetorna6(){
    assertEquals(new Integer("6"), profissionalAnalise.analisaMagnesio(101));
}
public void testAnalisaMagnesioRetorna7(){
    assertEquals(new Integer("7"), profissionalAnalise.analisaMagnesio(76));
}
public void testAnalisaMagnesioRetorna8(){
    assertEquals(new Integer("8"), profissionalAnalise.analisaMagnesio(51));
}
public void testAnalisaMagnesioRetorna9(){
    assertEquals(new Integer("9"), profissionalAnalise.analisaMagnesio(26));
}
public void testAnalisaMagnesioRetorna10(){
    assertEquals(new Integer("10"), profissionalAnalise.analisaMagnesio(11));
}
public void testAnalisaMagnesioRetorna11(){
    assertEquals(new Integer("11"), profissionalAnalise.analisaMagnesio(10));
}
```

Método: analisaFosfatos

Valor	Resultado
501	1
301	2
201	3
101	4
51	5
31	6
21	7
11	8
6	9
1	10
0	11

```
public void testAnalisaFosfatosRetorna1(){
    assertEquals(new Integer("1"), profissionalAnalise.analisaFosfatos(501));
}
public void testAnalisaFosfatosRetorna2(){
    assertEquals(new Integer("2"), profissionalAnalise.analisaFosfatos(301));
}
public void testAnalisaFosfatosRetorna3(){
    assertEquals(new Integer("3"), profissionalAnalise.analisaFosfatos(201));
}
public void testAnalisaFosfatosRetorna4(){
    assertEquals(new Integer("4"), profissionalAnalise.analisaFosfatos(101));
}
public void testAnalisaFosfatosRetorna5(){
    assertEquals(new Integer("5"), profissionalAnalise.analisaFosfatos(51));
}
public void testAnalisaFosfatosRetorna6(){
    assertEquals(new Integer("6"), profissionalAnalise.analisaFosfatos(31));
}
public void testAnalisaFosfatosRetorna7(){
    assertEquals(new Integer("7"), profissionalAnalise.analisaFosfatos(21));
}
public void testAnalisaFosfatosRetorna8(){
    assertEquals(new Integer("8"), profissionalAnalise.analisaFosfatos(11));
}
public void testAnalisaFosfatosRetorna9(){
    assertEquals(new Integer("9"), profissionalAnalise.analisaFosfatos(6));
}
public void testAnalisaFosfatosRetorna10(){
    assertEquals(new Integer("10"), profissionalAnalise.analisaFosfatos(1));
}
public void testAnalisaFosfatosRetorna11(){
    assertEquals(new Integer("11"), profissionalAnalise.analisaFosfatos(0));
}
```

Método: analisaNitritos

Valor	Resultado
1.01	1
0.51	2
0.26	3
0.21	4
0.16	5
0.11	6
0.06	7
0.026	8
0.011	9
0.001	10
0	11

```
public void testAnalisaNitritosRetorna1(){
    assertEquals(new Integer("1"), profissionalAnalise.analisaNitritos(1.01));
}
public void testAnalisaNitritosRetorna2(){
    assertEquals(new Integer("2"), profissionalAnalise.analisaNitritos(0.51));
}
public void testAnalisaNitritosRetorna3(){
    assertEquals(new Integer("3"), profissionalAnalise.analisaNitritos(0.26));
}
public void testAnalisaNitritosRetorna4(){
    assertEquals(new Integer("4"), profissionalAnalise.analisaNitritos(0.21));
}
public void testAnalisaNitritosRetorna5(){
    assertEquals(new Integer("5"), profissionalAnalise.analisaNitritos(0.16));
}
public void testAnalisaNitritosRetorna6(){
    assertEquals(new Integer("6"), profissionalAnalise.analisaNitritos(0.11));
}
public void testAnalisaNitritosRetorna7(){
    assertEquals(new Integer("7"), profissionalAnalise.analisaNitritos(0.06));
}
public void testAnalisaNitritosRetorna8(){
    assertEquals(new Integer("8"), profissionalAnalise.analisaNitritos(0.026));
}
public void testAnalisaNitritosRetorna9(){
    assertEquals(new Integer("9"), profissionalAnalise.analisaNitritos(0.011));
}
public void testAnalisaNitritosRetorna10(){
    assertEquals(new Integer("10"), profissionalAnalise.analisaNitritos(0.001));
}
public void testAnalisaNitritosRetorna11(){
    assertEquals(new Integer("11"), profissionalAnalise.analisaNitritos(0));
}
```

Método: analisaDbo

Valor	Resultado
16	1
13	2
11	3
9	4
7	5
6	6
5	7
3	8
2	9
0.6	10
0.5	11

```
public void testAnalisaDboRetorna1(){
    assertEquals(new Integer("1"), profissionalAnalise.analisaDbo(16));
}
public void testAnalisaDboRetorna2(){
    assertEquals(new Integer("2"), profissionalAnalise.analisaDbo(13));
}
public void testAnalisaDboRetorna3(){
    assertEquals(new Integer("3"), profissionalAnalise.analisaDbo(11));
}
public void testAnalisaDboRetorna4(){
    assertEquals(new Integer("4"), profissionalAnalise.analisaDbo(9));
}
public void testAnalisaDboRetorna5(){
    assertEquals(new Integer("5"), profissionalAnalise.analisaDbo(7));
}
public void testAnalisaDboRetorna6(){
    assertEquals(new Integer("6"), profissionalAnalise.analisaDbo(6));
}
public void testAnalisaDboRetorna7(){
    assertEquals(new Integer("7"), profissionalAnalise.analisaDbo(5));
}
public void testAnalisaDboRetorna8(){
    assertEquals(new Integer("8"), profissionalAnalise.analisaDbo(3));
}
public void testAnalisaDboRetorna9(){
    assertEquals(new Integer("9"), profissionalAnalise.analisaDbo(2));
}
public void testAnalisaDboRetorna10(){
    assertEquals(new Integer("10"), profissionalAnalise.analisaDbo(0.6));
}
public void testAnalisaDboRetorna11(){
    assertEquals(new Integer("11"), profissionalAnalise.analisaDbo(0.5));
}
```

Método: analisaCor

Valor	Resultado
251	1
101	2
61	3
41	4
31	5
21	6
16	7
11	8
6	9
4	10
3	11

```
public void testAnalisaCorRetorna1(){
    assertEquals(new Integer("1"), profissionalAnalise.analisaCor(251));
}
public void testAnalisaCorRetorna2(){
    assertEquals(new Integer("2"), profissionalAnalise.analisaCor(101));
}
public void testAnalisaCorRetorna3(){
    assertEquals(new Integer("3"), profissionalAnalise.analisaCor(61));
}
public void testAnalisaCorRetorna4(){
    assertEquals(new Integer("4"), profissionalAnalise.analisaCor(41));
}
public void testAnalisaCorRetorna5(){
    assertEquals(new Integer("5"), profissionalAnalise.analisaCor(31));
}
public void testAnalisaCorRetorna6(){
    assertEquals(new Integer("6"), profissionalAnalise.analisaCor(21));
}
public void testAnalisaCorRetorna7(){
    assertEquals(new Integer("7"), profissionalAnalise.analisaCor(16));
}
public void testAnalisaCorRetorna8(){
    assertEquals(new Integer("8"), profissionalAnalise.analisaCor(11));
}
public void testAnalisaCorRetorna9(){
    assertEquals(new Integer("9"), profissionalAnalise.analisaCor(6));
}
public void testAnalisaCorRetorna10(){
    assertEquals(new Integer("10"), profissionalAnalise.analisaCor(4));
}
public void testAnalisaCorRetorna11(){
    assertEquals(new Integer("11"), profissionalAnalise.analisaCor(3));
}
```

Método: analisaTurbidez

Valor	Resultado
401	1
251	2
181	3
101	4
51	5
21	6
16	7
11	8
6	9
4	10
3	11

```
public void testAnalisaTurbidezRetorna1(){
    assertEquals(new Integer("1"), profissionalAnalise.analisaTurbidez(401));
}

public void testAnalisaTurbidezRetorna2(){
    assertEquals(new Integer("2"), profissionalAnalise.analisaTurbidez(251));
}

public void testAnalisaTurbidezRetorna3(){
    assertEquals(new Integer("3"), profissionalAnalise.analisaTurbidez(181));
}

public void testAnalisaTurbidezRetorna4(){
    assertEquals(new Integer("4"), profissionalAnalise.analisaTurbidez(101));
}

public void testAnalisaTurbidezRetorna5(){
    assertEquals(new Integer("5"), profissionalAnalise.analisaTurbidez(51));
}

public void testAnalisaTurbidezRetorna6(){
    assertEquals(new Integer("6"), profissionalAnalise.analisaTurbidez(21));
}

public void testAnalisaTurbidezRetorna7(){
    assertEquals(new Integer("7"), profissionalAnalise.analisaTurbidez(16));
}

public void testAnalisaTurbidezRetorna8(){
    assertEquals(new Integer("8"), profissionalAnalise.analisaTurbidez(11));
}

public void testAnalisaTurbidezRetorna9(){
    assertEquals(new Integer("9"), profissionalAnalise.analisaTurbidez(6));
}

public void testAnalisaTurbidezRetorna10(){
    assertEquals(new Integer("10"), profissionalAnalise.analisaTurbidez(4));
}

public void testAnalisaTurbidezRetorna11(){
    assertEquals(new Integer("11"), profissionalAnalise.analisaTurbidez(3));
}
```

Método: analisaSodio

Valor	Resultado
501	1
301	2
251	3
201	4
151	5
101	6
76	7
51	8
26	9
11	10
10	11

```
public void testAnalisaSodioRetorna1(){
    assertEquals(new Integer("1"), profissionalAnalise.analisaSodio(501));
}
public void testAnalisaSodioRetorna2(){
    assertEquals(new Integer("2"), profissionalAnalise.analisaSodio(301));
}
public void testAnalisaSodioRetorna3(){
    assertEquals(new Integer("3"), profissionalAnalise.analisaSodio(251));
}
public void testAnalisaSodioRetorna4(){
    assertEquals(new Integer("4"), profissionalAnalise.analisaSodio(201));
}
public void testAnalisaSodioRetorna5(){
    assertEquals(new Integer("5"), profissionalAnalise.analisaSodio(151));
}
public void testAnalisaSodioRetorna6(){
    assertEquals(new Integer("6"), profissionalAnalise.analisaSodio(101));
}
public void testAnalisaSodioRetorna7(){
    assertEquals(new Integer("7"), profissionalAnalise.analisaSodio(76));
}
public void testAnalisaSodioRetorna8(){
    assertEquals(new Integer("8"), profissionalAnalise.analisaSodio(51));
}
public void testAnalisaSodioRetorna9(){
    assertEquals(new Integer("9"), profissionalAnalise.analisaSodio(26));
}
public void testAnalisaSodioRetorna10(){
    assertEquals(new Integer("10"), profissionalAnalise.analisaSodio(11));
}
public void testAnalisaSodioRetorna11(){
    assertEquals(new Integer("11"), profissionalAnalise.analisaSodio(10));
}
```

Método: analisaCalcio

Valor	Resultado
1001	1
601	2
501	3
401	4
301	5
201	6
151	7
101	8
51	9
11	10
10	11

```
public void testAnalisaCalcioRetorna1(){
    assertEquals(new Integer("1"), profissionalAnalise.analisaCalcio(1001));
}
public void testAnalisaCalcioRetorna2(){
    assertEquals(new Integer("2"), profissionalAnalise.analisaCalcio(601));
}
public void testAnalisaCalcioRetorna3(){
    assertEquals(new Integer("3"), profissionalAnalise.analisaCalcio(501));
}
public void testAnalisaCalcioRetorna4(){
    assertEquals(new Integer("4"), profissionalAnalise.analisaCalcio(401));
}
public void testAnalisaCalcioRetorna5(){
    assertEquals(new Integer("5"), profissionalAnalise.analisaCalcio(301));
}
public void testAnalisaCalcioRetorna6(){
    assertEquals(new Integer("6"), profissionalAnalise.analisaCalcio(201));
}
public void testAnalisaCalcioRetorna7(){
    assertEquals(new Integer("7"), profissionalAnalise.analisaCalcio(151));
}
public void testAnalisaCalcioRetorna8(){
    assertEquals(new Integer("8"), profissionalAnalise.analisaCalcio(101));
}
public void testAnalisaCalcioRetorna9(){
    assertEquals(new Integer("9"), profissionalAnalise.analisaCalcio(51));
}
public void testAnalisaCalcioRetorna10(){
    assertEquals(new Integer("10"), profissionalAnalise.analisaCalcio(11));
}
public void testAnalisaCalcioRetorna11(){
    assertEquals(new Integer("11"), profissionalAnalise.analisaCalcio(10));
}
```

6 – Mock

Diversos métodos não foram testados, entre eles estão os métodos de interação entre a página web, requisições, e o banco de dados. Para os métodos de Update foram criados objetos Mock para realização de teste.

Para implementação foi adicionado um retorno booleano para os métodos, implicando no sucesso ou não da operação.

Método: updateContrato

Antes

Contrato.class

```
public void updateContrato(HttpServletRequest request) throws SQLException, ClassNotFoundException{
    this.codigo = Integer.parseInt(request.getParameter("textCodigo"));
    this.local = Contrato.obterContrato(codigo).local;
    setParameter(request);
    ContratoDao.getInstance().update(this);
}
```

ContratoDao.class

```
public void update(Contrato contrato) throws SQLException, ClassNotFoundException{
    Connection conn = null;
    PreparedStatement stmt = null;

    try{
        conn = DatabaseLocator.getInstance().getConnection();
        stmt = conn.prepareStatement(
            "UPDATE CONTRATO SET "
            + "COD_CLIENTE = ?, "
            + "COD_EMPRESA = ?, "
            + "COD CONTRATO ESTADO = ?, "
            + "COD_LOCAL = ?, "
            + "DESCRICAO = ? WHERE CODIGO = ?");
        parseAtributos(stmt, contrato);
    } catch(SQLException e){
        throw e;
    } finally{
        closeResources(conn, stmt);
    }
}
```

Depois

Contrato.class

```
public Boolean updateContrato(HttpServletRequest request) throws SQLException, ClassNotFoundException{  
    this.codigo = Integer.parseInt(request.getParameter("textCodigo"));  
    this.local = Contrato.obterContrato(codigo).local;  
    setParameter(request);  
    return ContratoDao.getInstance().update(this);  
}
```

ContratoDao.class

```
public boolean update(Contrato contrato) throws SQLException, ClassNotFoundException{  
  
    Connection conn = null;  
    PreparedStatement stmt = null;  
    Boolean retorno = true;  
  
    try{  
        conn = DatabaseLocator.getInstance().getConnection();  
        stmt = conn.prepareStatement(  
            "UPDATE CONTRATO SET "  
            + "COD_CLIENTE = ?, "  
            + "COD_EMPRESA = ?, "  
            + "COD_CONTRATO_ESTADO = ?, "  
            + "COD_LOCAL = ?, "  
            + "|DESCRICAO = ? WHERE CODIGO = ?");  
        parseAtributos(stmt, contrato);  
  
    }catch(SQLException e){  
        retorno = false;  
        throw e;  
    }finally{  
        closeResources(conn, stmt);  
    }  
    return retorno;  
}
```

Teste

```
public void testContratoSucessoUpdate() throws SQLException, ClassNotFoundException {  
  
    HttpServletRequest requestMock = createMock(HttpServletRequest.class);  
    expect(requestMock.getParameter("textCodigo")).andReturn("17");  
    expect(requestMock.getParameter("textCodigoCliente")).andReturn("4");  
    expect(requestMock.getParameter("textCodigoEmpresa")).andReturn("3");  
    expect(requestMock.getParameter("textContratoEstado")).andReturn("1");  
    expect(requestMock.getParameter("textDescricao")).andReturn("0");  
    replay(requestMock);  
  
    Contrato contrato = new Contrato();  
    assertTrue(contrato.updateContrato(requestMock));  
}
```

Método: updateFuncionario

Antes

Funcionario.class

```
public void updateFuncionario(HttpServletRequest request) throws SQLException, ClassNotFoundException{  
    this.codigo = Integer.parseInt(request.getParameter("textCodigo"));  
    setParameter(request);  
    FuncionarioDao.getInstance().update(this);  
}
```

FuncionarioDao.class

```
public void update(Funcionario funcionario) throws SQLException, ClassNotFoundException{  
  
    Connection conn = null;  
    PreparedStatement stmt = null;  
  
    try{  
        conn = DatabaseLocator.getInstance().getConnection();  
        stmt = conn.prepareStatement("UPDATE FUNCIONARIO SET "  
            + "NOME = ?, "  
            + "IDENTIFICACAO = ?, "  
            + "COD_EMPRESA = ?, "  
            + "CARGO = ? WHERE CODIGO = ?");  
        parseAtributos(stmt, funcionario);  
  
    }catch(SQLException e){  
        throw e;  
    }finally{  
        closeResources(conn, stmt);  
    }  
}
```

Depois

Funcionario.class

```
public boolean updateFuncionario(HttpServletRequest request) throws SQLException, ClassNotFoundException{  
    this.codigo = Integer.parseInt(request.getParameter("textCodigo"));  
    setParameter(request);  
    return FuncionarioDao.getInstance().update(this);  
}
```

FuncionarioDao.class

```
public boolean update(Funcionario funcionario) throws SQLException, ClassNotFoundException{

    Connection conn = null;
    PreparedStatement stmt = null;
    Boolean retorno = true;

    try{
        conn = DatabaseLocator.getInstance().getConnection();
        stmt = conn.prepareStatement("UPDATE FUNCIONARIO SET "
            + "NOME = ?, "
            + "IDENTIFICACAO = ?, "
            + "COD_EMPRESA = ?, "
            + "CARGO = ? WHERE CODIGO = ?");
        parseAtributos(stmt, funcionario);

    }catch(SQLException e){
        retorno = false;
        throw e;
    }finally{
        closeResources(conn, stmt);
    }
    return retorno;
}
```

Teste

```
public void testFuncionarioSucessoUpdate() throws SQLException, ClassNotFoundException {

    HttpServletRequest requestMock = createMock(HttpServletRequest.class);
    expect(requestMock.getParameter("textCodigo")).andReturn("2");
    expect(requestMock.getParameter("textNome")).andReturn("alterado");
    expect(requestMock.getParameter("textIdentificacao")).andReturn("12345678910");
    expect(requestMock.getParameter("textEmpresa")).andReturn("3");
    expect(requestMock.getParameter("textCargo")).andReturn("1");
    replay(requestMock);

    Funcionario funcionario = new Funcionario();
    assertTrue(funcionario.updateFuncionario(requestMock));
}
```

7 – Cobertura dos testes criados

Dentre os testes unitários com e sem Mock segue a apresentação de cobertura destes gerada a partir do plugin JaCoverage.

model

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
C_Contrato		44%		31%	35	52	59	108	11	28	0	1
C_ProfissionalAnalise		90%		91%	25	290	48	553	2	27	0	1
C_Amostra		0%		0%	21	21	44	44	16	16	1	1
C_Empresa		27%	n/a	15	28	40	64	15	28	0	1	
C_Cliente		29%	n/a	11	20	28	45	11	20	0	1	
C_Local		32%	n/a	10	20	25	44	10	20	0	1	
C_Analise		0%	n/a	17	17	32	32	17	17	1	1	
C_Servico		0%	n/a	12	12	25	25	12	12	1	1	
C_ProfissionalEspecialista		0%		0%	8	8	16	16	3	3	1	1
C_Funcionario		50%	n/a	10	18	24	39	10	18	0	1	
C_ContratoServico		0%	n/a	11	11	21	21	11	11	1	1	
C_Resultado		0%	n/a	7	7	12	12	7	7	1	1	
C_ContratoEstadoAberto		0%	n/a	8	8	8	8	8	8	1	1	
C_ContratoEstadoEmColeta		0%	n/a	8	8	8	8	8	8	1	1	
C_ContratoEstadoAnaliseFinalizada		0%	n/a	8	8	8	8	8	8	1	1	
C_ContratoEstadoEmAnalise		0%	n/a	8	8	8	8	8	8	1	1	
C_ContratoEstadoColetaFinalizada		0%	n/a	8	8	8	8	8	8	1	1	
C_ContratoEstadoFechado		29%	n/a	6	8	6	8	6	8	0	1	
C_ContratoEstadoFactory		79%	50%	2	3	3	13	1	2	0	1	
C_ProfissionalColeta		0%	n/a	2	2	2	2	2	2	1	1	
Total	1.291 of 3.423	62%	100 of 596	83%	232	557	425	1.066	174	259	12	20

8 - Sellenium

Uma funcionalidade do sistema que não foi testada diretamente é a calculadora do sistema. Esta pela devida importância, foi escolhida como objetivo de teste da ferramenta Sellenium.

Segue abaixo a análise do valor limite e representação através de imagem dos testes ocorrendo.

TestCalculadoraMuitoRuim

Servico	valor	analiseServico
Ph	1	0
Condutividade	16001	0
OxigenioDissolvido	0.9	0
ReducaoPermanganato	16	0
ColiformesTotais	14001	0
NitrogenioAmoniacal	1.26	0
Cloretos	1501	0
Detergentes	4	0
DurezaAlcalinidade	1501	0
SolidosDissolvidos	20001	0
Praguicidas	2.1	0
GraxasAzeites	3.1	0
Sulfatos	151	0.5
Nitratos	9	0.5
Cianetos	0.21	0.5
CoLivre	11	0.5
Magnesio	101	0.5
Fosfatos	31	0.5
Nitritos	0.11	0.5
Dbo	5.1	0.5
Cor	21	0.5
Turbidez	21	0.5
Sodio	101	0.5
Calcio	201	0.5
Resultado Final		0.25
Resultado Qualitativo		Muito Ruim

The screenshot shows two windows side-by-side. On the left is a web browser window titled 'AGUA QUALIDADE' displaying a form with various water quality parameters and their values. On the right is the 'Selenium IDE - Untitled Project' interface, which is running a test script against the same URL.

Water Quality Parameters (Left Window):

- Graxas e Azeites: 3.1
- Sulfatos: 151
- Nitratos: 9
- Cianetos: 0.21
- CO2 Livre: 11
- Magnesio: 101
- Fosfatos: 31
- Nitritos: 0.11
- DBO: 5.1
- Cor: 21
- Turbidez: 21
- Sodio: 101
- Calcio: 1

Selenium IDE Test Script (Right Window):

```

Tests - + | D-E | □ | □ | II | ○ | 
Search tests... | 
Boa | 
Excelente | 
Media | 
MuitoRuim | 
Ruim | 
| Command | Target | Value |
| 17. type | id=10 | 20001 |
| 18. type | id=11 | 2.1 |
| 19. type | id=12 | 3.1 |
| 20. type | id=13 | 151 |
| 21. type | id=14 | 9 |
| 22. type | id=15 | 0.21 |
| 23. type | id=16 | 11 |
| 24. type | id=17 | 101 |
| 25. type | id=18 | 31 |
| 26. type | id=19 | 0.11 |
| 27. type | id=20 | 5.1 |
| 28. type | id=21 | 21 |
| 29. type | id=22 | 21 |
| 30. type | id=23 | 101 |
| 31. type | id=24 | 201 |
| 32. click at | //input[@value='Gravar'] | 327.17 |
| 33. assert value | xpath=/input[@name='textCodigo'][2] | Muito Ruim |
| 
| Command | open |
| Target | http://localhost:8080/ControleQualidadeAgua |
| Value | |
| Comment | |
| Runs: 0 Failures: 0 |
| Log |
| 25. Trying to execute type on id=19 with value 31... Success |
| 26. Trying to execute type on id=19 with value 0.11... Success |
| 27. Trying to execute type on id=20 with value 5.1... Success |
| 28. Trying to execute type on id=21 with value 21... Success |
| 29. Trying to execute type on id=22 with value 21... Success |
| 30. Trying to execute type on id=23 with value 101... Success |
| 31. Trying to execute type on id=24 with value 201... Success |
| 

```

TestCalculadoraRuim

Servico	valor	analiseServico
Ph	6	0.5
Condutividade	2501	0.5
OxigenioDissolvido	4.9	0.5
ReducaoPermanganato	5.1	0.5
ColiformesTotais	3001	0.5
NitrogenioAmoniacal	0.31	0.5
Cloretos	201	0.5
Detergentes	0.51	0.5
DurezaAlcalinidade	401	0.5
SolidosDissolvidos	1501	0.5
Praguicidas	0.06	0.5
GraxasAzeites	0.16	0.5
Sulfatos	151	0.5
Nitratos	9	0.5

Cianetos	0.21	0.5
CoLivre	11	0.5
Magnesio	101	0.5
Fosfatos	31	0.5
Nitritos	0.11	0.5
Dbo	5.1	0.5
Cor	21	0.5
Turbidez	21	0.5
Sodio	101	0.5
Calcio	201	0.5
Resultado Final		0.5
Resultado Qualitativo		Ruim

The screenshot shows a web browser window titled "IA QUALIDADE" displaying a result dialog. The dialog is titled "Resultado IQA" and contains a red "X" icon at the top right. It has a single input field labeled "Resultado" containing the value "Ruim". Below the input field is a green "OK" button. Above the dialog, the browser's address bar shows the URL "localhost:8080/ControleQualidadeAgua/FrontController?action=CalcularCalculadora". A tooltip indicates that "NetBeans Connector" is debugging the browser.

To the right of the browser is the NetBeans IDE interface. A window titled "Untitled Project" is open, showing a table of test cases under the heading "Default Suite". The table includes columns for "Command", "Target", and "Value". One row is highlighted with a blue background and the value "Ruim" in the "Value" column. The "Command" column contains various Selenium-like commands such as "type", "click at", and "assert value". The "Target" column lists URLs like "http://localhost:8080/ControleQualidadeAgua". At the bottom of the test runner window, it says "Runs: 5 Failures: 0".

TestCalculadoraMedia

Servico	valor	analiseServico
Ph	8	0.7
Condutividade	1501	0.7
OxigenioDissolvido	6.4	0.7

ReducaoPermanganato	3.1	0.7
ColiformesTotais	1501	0.7
NitrogenioAmoniacal	0.11	0.7
Cloretos	101	0.7
Detergentes	0.11	0.7
DurezaAlcalinidade	201	0.7
SolidosDissolvidos	751	0.7
Praguicidas	0.011	0.7
GraxasAzeites	0.05	0.7
Sulfatos	76	0.7
Nitratos	5	0.7
Cianetos	0.06	0.7
CoLivre	8.1	0.7
Magnesio	51	0.7
Fosfatos	11	0.7
Nitritos	0.026	0.7
Dbo	2.1	0.7
Cor	11	0.7
Turbidez	11	0.7
Sodio	51	0.7
Calcio	101	0.7
Resultado Final		0.7
Resultado Qualitativo		Media

The screenshot shows a web browser window titled "AGUA QUALIDADE" and a "Selenium IDE - Untitled Project" window side-by-side.

Web Browser (Left):

- URL: localhost:8080/ControleQualidadeAgua/FrontController?action=CarregarCalculadora
- Form fields (values):
 - Graxas e Azeites: 0.05
 - Sulfatos: 76
 - Nitratos: 5
 - Cianetos: 0.06
 - CO2 Livre: 8.1
 - Magnesio: 51
 - Fosfatos: 11
 - Nitritos: 0.026
 - DBO: 2.1
 - Cor: 11
 - Turbidez: 11

Selenium IDE (Right):

- Test title: Untitled Project
- Test description: "NetBeans Connector" está depurando esse navegador
- Test steps (highlighted in yellow):
 - 17. click at id=11
 - 18. type id=11
 - 19. type id=12
 - 20. type id=13
 - 21. type id=14
 - 22. type id=15
 - 23. type id=16
 - 24. type id=17
 - 25. type id=18
 - 26. type id=19
 - 27. type id=20
 - 28. type id=21
 - 29. type id=22
 - 30. type id=23
 - 31. type id=24
 - 32. click at //input[@value='Gravar']
 - 33. assert value xpath://input[@name='textCodigo'][2] = Media
- Test configuration:
 - Command: open
 - Target: http://localhost:8080/ControleQualidadeAgua
 - Value:
 - Comment:
- Log output:
 - 29. trying to execute type on id=id with value 0.026... Success
 - 30. trying to execute type on id=id with value 11... Success
 - 31. trying to execute type on id=id with value 11... Success
 - 32. trying to execute click on id=id with value Gravar... Success
 - 33. trying to execute assert value on xpath://input[@name='textCodigo'][2] = Media... Success

TestCalculadoraBoa

Servico	valor	analiseServico
Ph	10	0.9
Condutividade	750	0.9
OxigenioDissolvido	6.9	0.9
ReducaoPermanganato	0.6	0.9
ColiformesTotais	51	0.9
NitrogenioAmoniacal	0.01	0.9
Cloreto	1	0.9
Detergentes	0.01	0.9
DurezaAlcalinidade	26	0.9
SolidosDissolvidos	101	0.9
Praguicidas	0.001	0.9
GraxasAzeites	0.01	0.9
Sulfatos	1	0.9
Nitratos	1	0.9
Cianetos	0.01	0.9
CoLivre	4	0.9
Magnesio	11	0.9
Fosfatos	1	0.9

Nitritos	0.001	0.9
Dbo	0.6	0.9
Cor	4	0.9
Turbidez	4	0.9
Sodio	11	0.9
Calcio	11	0.9
Resultado Final		0.9
Resultado Qualitativo		Boa

The screenshot displays the Selenium IDE interface. On the left, there is a browser window titled "AGUA QUALIDADE" showing a form with various input fields for water quality parameters. On the right, the "Selenium IDE - Untitled Project" window is open, showing a test log and a command table.

Test Log:

```

22. Trying to execute type on id=10 with value 4... Success
23. Trying to execute type on id=17 with value 11... Success
24. Trying to execute type on id=18 with value 1... Success
25. Trying to execute type on id=19 with value 0.001... Success
26. Trying to execute type on id=20 with value 0.6... Success
27. Trying to execute type on id=21 with value 4... Success
28. Trying to execute type on id=22 with value 4... Success
29. Trying to execute type on id=23 with value 11... Success
30. Trying to execute type on id=24 with value 11... Success
    
```

Command Table:

Command	Target	Value
id=6	0.01	0.01
id=7	1	1
id=8	0.01	0.01
id=9	26	26
id=10	101	101
id=11	0.001	0.001
id=12	0.01	0.01
id=13	1	1
id=14	1	1
id=15	0.01	0.01
id=16	4	4
id=17	11	11
id=18	1	1
id=19	0.001	0.001
id=20	0.6	0.6
id=21	4	4
id=22	4	4
id=23	11	11
id=24	11	11

TestCalculadoraExcelente

Serviço	valor	analiseServico
Ph	9	0.8
Condutividade	1251	0.8
OxigenioDissolvido	6.9	0.8
ReducaoPermanganato	2.1	0.8
ColiformesTotais	1001	0.8
NitrogenioAmoniacal	0.01	0.9

Cloreto	1	0.9
Detergentes	0.01	0.9
Dureza Alcalinidade	26	0.9
Solidos Dissolvidos	101	0.9
Praguicidas	0.001	0.9
Graxas Azeites	0.01	0.9
Sulfatos	1	0.9
Nitratos	1	0.9
Cianetos	0.01	0.9
CoLivre	3	1
Magnesio	10	1
Fosfatos	0	1
Nitritos	0	1
Dbo	0.5	1
Cor	3	1
Turbidez	3	1
Sodio	10	1
Calcio	10	1
Resultado Final		0.91
Resultado Qualitativo		Excelente

AGUA QUALIDADE

localhost:8080/ControleQualidadeAgua/FrontController?action=CarregarCalculadora

"NetBeans Connector" está depurando esse navegador

Selenium IDE - Untitled Project

"NetBeans Connector" está depurando esse navegador

Untitled Project

Tests + http://localhost:8080

Command	Target	Value
type	id=4	2.1
type	id=5	1001
type	id=6	0.01
type	id=7	1
type	id=8	0.01
type	id=9	26
type	id=10	101
type	id=11	0.001
type	id=12	0.01
type	id=13	1
type	id=14	1
type	id=15	0.01
type	id=16	3
type	id=17	10
type	id=18	0
type	id=19	0
type	id=20	0.5
type	id=21	3
type	id=22	3

Command: open

Target: http://localhost:8080/ControleQualidadeAgua

Value:

Comment:

Runs: 0 Failures: 0

Log

```
19. Trying to execute type on id=13 with value 1... Success
20. Trying to execute type on id=14 with value 1... Success
21. Trying to execute type on id=15 with value 0.01... Success
22. Trying to execute type on id=16 with value 3... Success
23. Trying to execute type on id=17 with value 10... Success
24. Trying to execute type on id=18 with value 0... Success
25. Trying to execute type on id=19 with value 0... Success
26. Trying to execute type on id=20 with value 0.5...
```